



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.245

(09/98)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS
Infrastructure of audiovisual services – Communication
procedures

Control protocol for multimedia communication

ITU-T Recommendation H.245

(Previously CCITT Recommendation)

ITU-T H-SERIES RECOMMENDATIONS
AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Characteristics of transmission channels used for other than telephone purposes	H.10–H.19
Use of telephone-type circuits for voice-frequency telegraphy	H.20–H.29
Telephone circuits or cables used for various types of telegraph transmission or simultaneous transmission	H.30–H.39
Telephone-type circuits used for facsimile telegraphy	H.40–H.49
Characteristics of data signals	H.50–H.99
CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.399
Supplementary services for multimedia	H.450–H.499

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION H.245

CONTROL PROTOCOL FOR MULTIMEDIA COMMUNICATION

Source

ITU-T Recommendation H.245 was revised by ITU-T Study Group 16 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 25th of September 1998.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation the term *recognized operating agency (ROA)* includes any individual, company, corporation or governmental organization that operates a public correspondence service. The terms *Administration*, *ROA* and *public correspondence* are defined in the *Constitution of the ITU (Geneva, 1992)*.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

© ITU 1999

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the ITU.

CONTENTS

	Page
1 Scope	1
2 References	1
3 Definitions	4
4 Abbreviations	6
5 General	7
5.1 Master-slave determination	7
5.2 Capability exchange	8
5.3 Logical channel signalling procedures	8
5.4 Receive terminal close logical channel request.....	9
5.5 H.223 multiplex table entry modification	9
5.6 Audiovisual and data mode request	9
5.7 Round-trip delay determination.....	9
5.8 Maintenance loops.....	9
5.9 Commands and indications	9
Annex A – Messages: Syntax.....	10
Annex B – Messages: Semantic definitions	65
B.1 Master Slave Determination messages.....	66
B.1.1 Master Slave Determination.....	66
B.1.2 Master Slave Determination Acknowledge	66
B.1.3 Master Slave Determination Reject	66
B.1.4 Master Slave Determination Release	66
B.2 Terminal capability messages	66
B.2.1 Overview	66
B.2.2 Terminal Capability Set	67
B.2.3 Terminal Capability Set Acknowledge	89
B.2.4 Terminal Capability Set Reject	89
B.2.5 Terminal Capability Set Release.....	89
B.3 Logical channel signalling messages.....	89
B.3.1 Open Logical Channel.....	90
B.3.2 Open Logical Channel Acknowledge	94
B.3.3 Open Logical Channel Reject	96
B.3.4 Open Logical Channel Confirm.....	97
B.3.5 Close Logical Channel	97
B.3.6 Close Logical Channel Acknowledge.....	98

	Page
B.3.7 Request Channel Close	98
B.3.8 Request Channel Close Acknowledge	98
B.3.9 Request Channel Close Reject	98
B.3.10 Request Channel Close Release	98
B.4 Multiplex Table signalling messages	98
B.4.1 Multiplex Entry Send	99
B.4.2 Multiplex Entry Send Acknowledge	99
B.4.3 Multiplex Entry Send Reject	99
B.4.4 Multiplex Entry Send Release	100
B.5 Request Multiplex Table signalling messages	100
B.5.1 Request Multiplex Entry	100
B.5.2 Request Multiplex Entry Acknowledge	100
B.5.3 Request Multiplex Entry Reject	100
B.5.4 Request Multiplex Entry Release	100
B.6 Request Mode messages	100
B.6.1 Request Mode	100
B.6.2 Request Mode Acknowledge	104
B.6.3 Request Mode Reject	104
B.6.4 Request Mode Release	104
B.7 Round Trip Delay messages	104
B.7.1 Round Trip Delay Request	105
B.7.2 Round Trip Delay Response	105
B.8 Maintenance Loop messages	105
B.8.1 Maintenance Loop Request	105
B.8.2 Maintenance Loop Acknowledge	105
B.8.3 Maintenance Loop Reject	105
B.8.4 Maintenance Loop Command Off	105
B.9 Communication Mode Messages	105
B.9.1 Communication Mode Command	105
B.9.2 Communication Mode Request	106
B.9.3 Communication Mode Response	106
B.10 Conference Request and Response Messages	106
B.10.1 Terminal List Request	106
B.10.2 Terminal List Response	106
B.10.3 Make Me Chair	107
B.10.4 Cancel Make Me Chair	107
B.10.5 Make Me Chair Response	107
B.10.6 Drop Terminal	107

	Page
B.10.7 Terminal Drop Reject.....	107
B.10.8 RequestTerminal ID.....	107
B.10.9 MC Terminal ID Response	107
B.10.10 Enter H.243 Password Request.....	107
B.10.11 Password Response	107
B.10.12 Enter H.243 Terminal ID Request	107
B.10.13 Terminal ID Response	107
B.10.14 Enter H.243 Conference ID Request.....	107
B.10.15 Conference ID Response.....	107
B.10.16 Video Command Reject.....	107
B.10.17 Enter Extension Address Request.....	107
B.10.18 Extension Address Response	107
B.10.19 Request Chair Control Token Owner.....	108
B.10.20 Chair Token Owner Response	108
B.10.21 Request Terminal Certificate	108
B.10.22 Terminal Certificate Response.....	108
B.10.23 Broadcast My Logical Channel.....	108
B.10.24 Broadcast My Logical Channel Response	108
B.10.25 Make Terminal Broadcaster.....	108
B.10.26 Make Terminal Broadcaster Response.....	109
B.10.27 Send This Source	109
B.10.28 SendThis Source Response	109
B.10.29 Request All Terminals Ids.....	109
B.10.30 Request All Terminal Ids Response.....	109
B.10.31 RemoteMC Request.....	109
B.10.32 RemoteMC Response.....	109
B.11 Multilink Messages	109
B.11.1 callInformation Request and Response.....	110
B.11.2 addConnection Request and Response	110
B.11.3 removeConnection Request and Response	110
B.11.4 maximumHeaderInterval Request and Response.....	110
B.11.5 Multilink Indications.....	111
B.11.6 DiallingInformation	111
B.11.7 DiallingInformationNumber	111
B.11.8 DiallingInformationNetworkType	111
B.11.9 ConnectionIdentifier	111
B.11.10 New ATM Virtual Channel Command.....	112

	Page	
B.12	Commands.....	112
	B.12.1 Send Terminal Capability Set	112
	B.12.2 Encryption.....	113
	B.12.3 Flow Control	113
	B.12.4 End session.....	113
	B.12.5 Miscellaneous Command.....	114
	B.12.6 Conference Command.....	116
	B.12.7 H.223 Multiplex Reconfiguration	117
B.13	Indications	117
	B.13.1 Function Not Understood.....	117
	B.13.2 Miscellaneous Indication	117
	B.13.3 Jitter Indication	118
	B.13.4 H.223 Skew Indication.....	119
	B.13.5 New ATM Virtual Channel Indication	119
	B.13.6 User Input.....	120
	B.13.7 Conference Indications.....	122
	B.13.8 H2250 Maximum Logical Channel Skew.....	122
	B.13.9 MC Location Indication	123
	B.13.10 Vendor Identification Indication	123
	B.13.11 Function Not Supported.....	123
Annex C	– Procedures	123
C.1	Introduction	123
	C.1.1 Method of specification	124
	C.1.2 Communication between protocol entity and protocol user	124
	C.1.3 Peer-to-peer communication	124
	C.1.4 SDL diagrams	124
	C.1.5 SDL Key.....	125
C.2	Master slave determination procedures.....	126
	C.2.1 Introduction.....	126
	C.2.2 Communication between the MSDSE and the MSDSE user.....	127
	C.2.3 Peer-to-peer MSDSE communication.....	129
	C.2.4 MSDSE procedures.....	130
C.3	Capability exchange procedures.....	136
	C.3.1 Introduction.....	136
	C.3.2 Communication between CESE and CESE user	137
	C.3.3 Peer-to-peer CESE communication	140
	C.3.4 CESE procedures	141

	Page
C.4 Unidirectional Logical Channel signalling procedures	144
C.4.1 Introduction	144
C.4.2 Communication between the LCSE and the LCSE user	146
C.4.3 Peer-to-peer LCSE communication	149
C.4.4 LCSE procedures	150
C.5 Bidirectional Logical Channel signalling procedures	158
C.5.1 Introduction	158
C.5.2 Communication between the B-LCSE and the B-LCSE user	160
C.5.3 Peer-to-peer B-LCSE communication	163
C.5.4 B-LCSE procedures	164
C.6 Close Logical Channel procedures	173
C.6.1 Introduction	173
C.6.2 Communication between CLCSE and CLCSE user	173
C.6.3 Peer-to-peer CLCSE communication	175
C.6.4 CLCSE procedures	176
C.7 H.223 multiplex table procedures	181
C.7.1 Introduction	181
C.7.2 Communication between the MTSE and MTSE user	183
C.7.3 Peer-to-peer MTSE communication	185
C.7.4 MTSE procedures	186
C.8 Request Multiplex Entry procedures	192
C.8.1 Introduction	192
C.8.2 Communication between RMESE and RMESE user	193
C.8.3 Peer-to-peer RMESE communication	195
C.8.4 RMESE procedures	196
C.9 Mode Request procedures	200
C.9.1 Introduction	200
C.9.2 Communication between MRSE and MRSE user	202
C.9.3 Peer-to-peer MRSE communication	204
C.9.4 MRSE procedures	204
C.10 Round trip delay procedures	210
C.10.1 Introduction	210
C.10.2 Communication between the RTDSE and the RTDSE user	211
C.10.3 Peer-to-peer RTDSE communication	212
C.10.4 RTDSE procedures	213

	Page
C.11 Maintenance Loop procedures	216
C.11.1 Introduction.....	216
C.11.2 Communication between the MLSE and the MLSE user	217
C.11.3 Peer-to-peer MLSE communication	220
C.11.4 MLSE procedures	221
Annex D – Object identifier assignments.....	228
Appendix I – Overview of ASN.1 syntax.....	228
I.1 Introduction to ASN.1	228
I.2 Basic ASN.1 data types	229
I.3 Aggregate data types	230
I.4 Object Identifier type.....	231
Appendix II – Examples of H.245 procedures	232
II.1 Introduction	232
II.2 Master Slave Determination Signalling Entity.....	233
II.3 Capability Exchange Signalling Entity	237
II.4 Logical Channel Signalling Entity	239
II.5 Close Logical Channel Signalling Entity	242
II.6 Multiplex Table Signalling Entity.....	243
II.7 Mode Request Signalling Entity.....	245
II.8 Round Trip Delay Signalling Entity.....	247
II.9 Bidirectional Logical Channel Signalling Entity	249
Appendix III – Summary of procedure timers and counters.....	251
III.1 Timers	251
III.2 Counters	252
Appendix IV – H.245 Extension Procedure	252
Appendix V – The replacementFor Procedure	254
Appendix VI – Examples of H.263 Capability Structure Settings	255
VI.1 Examples of Enhancement Layer H.245 parameter setting	255
VI.2 Examples of Video Back Channel H.245 parameter setting.....	257

Recommendation H.245

CONTROL PROTOCOL FOR MULTIMEDIA COMMUNICATION

(revised in 1998)

1 Scope

This Recommendation specifies syntax and semantics of terminal information messages as well as procedures to use them for in-band negotiation at the start of or during communication. The messages cover receiving and transmitting capabilities as well as mode preference from the receiving end, logical channel signalling, and Control and Indication. Acknowledged signalling procedures are specified to ensure reliable audiovisual and data communication.

This Recommendation covers a wide range of applications, including storage/retrieval, messaging and distribution services as well as conversational. It applies to, but is not limited to, multimedia systems that use the multiplexes defined in Recommendations H.222.0, H.223 and H.225.0. These different systems share the same syntax and semantics, and are therefore bit-wise compatible. Some of the procedures are applicable to all systems, while the others are more specific to particular systems.

The different systems that make use of this Recommendation may specify the use of different transport protocols. However, it is intended to be used with a reliable transport layer, that is, one that provides guaranteed delivery of correct data.

NOTE – There should be no confusion with the T.120 management system, which is carried within the data stream, and covers different functionalities from those described here – the H.245 stream and the T.120-data stream are complementary.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

- [1] ITU-T Recommendation E.164 (1997), *The international public telecommunication numbering plan.*
- [2] CCITT Recommendation G.711 (1988), *Pulse Code Modulation (PCM) of voice frequencies.*
- [3] CCITT Recommendation G.722 (1988), *7 kHz audio-coding within 64 kbit/s.*
- [4] ITU-T Recommendation G.723.1 (1996), *Speech coders: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s.*
- [5] CCITT Recommendation G.728 (1992), *Coding of speech at 16 kbit/s using low-delay code excited linear prediction.*
- [6] ITU-T Recommendation G.729 (1996), *Coding of speech at 8 kbit/s using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP).*

- [7] ITU-T Recommendation H.221 (1997), *Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices.*
- [8] ITU-T Recommendation H.222.0 (1995) | ISO/IEC 13818-1:1996, *Information technology – Generic coding of moving pictures and associated audio information: Systems.*
- [9] ITU-T Recommendation H.222.1 (1996), *Multimedia multiplex and synchronization for audiovisual communication in ATM environments.*
- [10] ITU-T Recommendation H.223 (1996), *Multiplexing protocol for low bit rate multimedia communication.*
- [11] ITU-T Recommendation H.224 (1994), *A real time control protocol for simplex applications using the H.221 LSD/HSD/MLP channels.*
- [12] ITU-T Recommendation H.225.0 (1998), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems.*
- [13] ITU-T Recommendation H.230 (1997), *Frame-synchronous control and indication signals for audiovisual systems.*
- [14] ITU-T Recommendation H.233 (1995), *Confidentiality system for audiovisual services.*
- [15] ITU-T Recommendation H.234 (1994), *Encryption key management and authentication system for audiovisual services.*
- [16] ITU-T Recommendation H.235 (1998), *Security and encryption for H-series (H.323 and other H.245 based) multimedia terminals.*
- [17] ITU-T Recommendation H.243 (1997), *Procedures for establishing communication between three or more audiovisual terminals using digital channels up to 1920 kbit/s.*
- [18] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at $p \times 64$ kbit/s.*
- [19] ITU-T Recommendation H.262 (1995) | ISO/IEC 13818-2:1996, *Information technology – Generic coding of moving pictures and associated audio information: Video.*
- [20] ITU-T Recommendation H.263 (1998), *Video coding for low bit rate communication.*
- [21] ITU-T Recommendation H.310 (1998), *Broadband audiovisual communication systems and terminals.*
- [22] ITU-T Recommendation H.320 (1997), *Narrow-band visual telephone systems and terminal equipment.*
- [23] ITU-T Recommendation H.323 (1998), *Packet-based multimedia communications systems.*
- [24] ITU-T Recommendation H.324 (1998), *Terminal for low bit-rate multimedia communication.*
- [25] ITU-T Recommendation I.363 (1996), *B-ISDN ATM adaptation layer specification.*
- [26] ITU-T Recommendation Q.2931 (1995), *Digital Subscriber Signalling System No. 2 – User-Network Interface (UNI) layer 3 specification for basic call/connection control.*
- [27] ITU-T Recommendation T.30 (1996), *Procedures for document facsimile transmission in the general switched telephone network.*
- [28] CCITT Recommendation T.35 (1991), *Procedure for the allocation of CCITT defined codes for non-standard facilities.*
- [29] ITU-T Recommendation T.38 (1998), *Procedures for real time Group 3 facsimile communication over IP Networks.*

- [30] CCITT Recommendation T.51 (1992), *Latin based coded character sets for telematic services.*
- [31] ITU-T Recommendation T.84 (1996) | ISO/IEC 10918-3:1996, *Information technology – Digital compression and coding of continuous-tone still images: Extensions.*
- [32] ITU-T Recommendation T.120 (1996), *Data protocols for multimedia conferencing.*
- [33] ITU-T Recommendation T.123 (1996), *Network specific data protocol stacks for multimedia conferencing.*
- [34] ITU-T Recommendation T.434 (1996), *Binary file transfer format for the telematic services.*
- [35] ITU-T Recommendation V.14 (1993), *Transmission of start-stop characters over synchronous bearer channels.*
- [36] ITU-T Recommendation V.34 (1998), *A modem operating at data signalling rates of up to 33 600 bit/s for use on the general switched telephone network and on leased point-to-point 2-wire telephone-type circuits.*
- [37] ITU-T Recommendation V.42 (1996), *Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion.*
- [38] ITU-T Recommendation X.680 (1994), *Information technology – Abstract Syntax Notation One (ASN.1) – Specification of basic notation.*
- [39] ITU-T Recommendation X.680/Amd.1 (1995), *Information technology – Abstract Syntax Notation One (ASN.1) – Specification of Basic Notation – Amendment 1: Rules of extensibility.*
- [40] ITU-T Recommendation X.681/Amd.1 (1995), *Information technology – Abstract Syntax Notation One (ASN.1) – Information Object Specification – Amendment 1: Rules of extensibility.*
- [41] ITU-T Recommendation X.691 (1997), *Information technology – ASN.1 Encoding Rules – Specification of Packed Encoding Rules (PER).*
- [42] ISO/IEC 3309:1993, *Information technology – Telecommunications and information exchange between systems – High-level Data Link Control (HDLC) procedures – Frame structure.*
- [43] ISO/IEC 11172-2:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 2: Video.*
- [44] ISO/IEC 11172-3:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 3: Audio.*
- [45] ISO/IEC 13818-3:1995, *Information technology – Generic coding of moving pictures and associated audio – Part 3: Audio.*
- [46] ISO/IEC 13818-6:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 6: Extensions for DSM-CC.*
- [47] ISO/IEC TR 9577:1996, *Information technology – Protocol identification in the network layer.*
- [48] I-ETS Recommendation 300 036 (GSM 06.10), *Full rate speech transcoding.*
- [49] I-ETS Recommendation 300 969 (GSM 06.20), *Half rate speech transcoding.*
- [50] I-ETS Recommendation 300 726 (GSM 06.60), *Enhanced full rate (EFR) speech transcoding.*

- [51] I-ETS Recommendation 300 038 (GSM 06.12), *Comfort noise aspects for full rate speech traffic channels.*
- [52] I-ETS Recommendation 300 971 (GSM 06.22), *Comfort noise aspects for half rate speech traffic channels.*
- [53] I-ETS Recommendation 300 728 (GSM 06.62), *Comfort noise aspects for enhanced full rate speech traffic channels.*
- [54] I-ETS Recommendation 300 039 (GSM 06.31), *Discontinuous transmission for full rate speech traffic channels.*
- [55] I-ETS Recommendation 300 972 (GSM 06.41), *Discontinuous transmission for half rate speech traffic channels.*
- [56] I-ETS Recommendation 300 729 (GSM 06.81), *Discontinuous transmission for enhanced full rate speech traffic channels.*
- [57] I-ETS Recommendation 300 037 (GSM 06.11), *Substitution and muting of lost frames for full rate speech traffic channels.*
- [58] I-ETS Recommendation 300 970 (GSM 06.21), *Substitution and muting of lost frames for half rate speech traffic channels.*
- [59] I-ETS Recommendation 300 727 (GSM 06.61), *Substitution and muting of lost frames for enhanced full rate speech traffic channels.*
- [60] I-ETS Recommendation 300 040 (GSM 06.32), *Voice activity detection for full rate speech traffic channels.*
- [61] I-ETS Recommendation 300 973 (GSM 06.42), *Voice activity detection for half rate speech traffic channels.*
- [62] I-ETS Recommendation 300 730 (GSM 06.82), *Voice activity detection for enhanced full rate speech traffic channels.*
- [63] I-ETS Recommendation 300 724 (GSM 06.53), *ANSI-C code for the GSM enhanced full rate speech.*

3 Definitions

This Recommendation uses the following terms.

3.1 bidirectional logical channel: A bidirectional logical channel consists of a pair of associated transmission paths between two terminals, one in each direction of transmission.

3.2 capability: A terminal has a particular capability if it is able to encode and transmit or receive and decode that particular signal.

3.3 channel: A channel is a unidirectional link between two end points.

3.4 command: A command is a message that requires action but no explicit response.

3.5 elementary stream: Elementary Stream is a generic term for a coded video, coded audio or other coded bitstream.

3.6 entry: The word entry is used to refer to elements in sets or tables, such as capability sets and multiplex tables.

3.7 forward: Forward is used to refer to transmission directed from the terminal making the request for a bidirectional logical channel to the other terminal.

- 3.8 in-band:** In-band messages are those that are transported within the channel or logical channel to which they refer.
- 3.9 incoming:** An Incoming Signalling Entity cannot initiate a procedure, but responds to messages from the remote Signalling Entity and its own user's primitives.
- 3.10 indication:** An indication is a message that contains information but does not require action or response.
- 3.11 logical channel:** A logical channel is a unidirectional path or bidirectional path for the transmission of information.
- 3.12 logical channel number:** A logical channel number is a number that identifies a single logical channel.
- 3.13 logical channel signalling:** Logical channel signalling is a set of procedures that are used to open and close logical channels.
- 3.14 master terminal:** A master terminal is the terminal that is determined as being the master terminal by the master-slave determination procedure defined in this Recommendation, or by some other procedure.
- 3.15 medium type:** A medium type is a single form of information that is presented to a user or the data representing that information: video, audio and text are example Medium Types.
- 3.16 mode:** A mode is a set of elementary streams that a terminal is transmitting, intends to transmit, or would like to receive.
- 3.17 multimedia communication:** Multimedia communication refers to the transmission and/or reception of signals of two or more Medium Types simultaneously.
- 3.18 multipoint:** Multipoint refers to the simultaneous interconnection of three or more terminals to allow communication among several sites through the use of multipoint control units (bridges) that centrally direct the flow of information.
- 3.19 non-standard:** Not conforming to a national or international standard referenced in this Recommendation.
- 3.20 outgoing:** An Outgoing Signalling Entity is one which initiates a procedure.
- 3.21 request:** A request is a message that results in action by the remote terminal and requires an immediate response from it.
- 3.22 response:** A response is a message that is the response to a request.
- 3.23 reverse:** Reverse is used to refer to transmission directed from the terminal receiving a request for a bidirectional logical channel to the terminal making the request.
- 3.24 session:** A session is a period of communication between two terminals which may be conversational or non-conversational (for example, retrieval from a database).
- 3.25 slave terminal:** A slave terminal is the terminal that is determined as being the slave terminal by the master-slave determination procedure defined in this Recommendation, or by some other procedure.
- 3.26 support:** The ability to operate in a given mode, however a requirement to support a mode does not mean that the mode must actually be used at all times: unless prohibited, other modes may be used by mutual negotiation.
- 3.27 terminal:** A terminal is any endpoint and may be a user's terminal or some other communication system such as an MCU or an information server.

3.28 TSAP identifier: The piece of information used to multiplex several transport connections of the same type on a single H.323 entity with all transport connections sharing the same LAN address, (e. g. the port number in a TCP/UDP/IP environment). TSAP identifiers may be (pre)assigned by some international authority or may be allocated dynamically during set-up of a call. Dynamically assigned TSAP identifiers are of transient nature, i. e. their values are only valid for the duration of a single call.

3.29 unidirectional logical channel: A unidirectional logical channel is a path for the transmission of a single elementary stream from one terminal to another.

4 Abbreviations

This Recommendation uses the following abbreviations:

AAL	ATM Adaptation Layer
AL1, 2, 3	H.223 Adaptation Layers 1, 2 and 3
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
B-LCSE	Bidirectional Logical Channel Signalling Entity
CESE	Capability Exchange Signalling Entity
CIF	Common Intermediate Format (of a video picture: refer to Recommendations H.261 and H.263)
CLCSE	Close Logical Channel Signalling Entity
CPCS	Common Part Convergence Sublayer (of ATM Adaptation Layer 5)
DSM-CC	Digital Storage Media-Command and Control
DTMF	Dual Tone Multi-Frequency
GOB	Group of Blocks (of a video picture: refer to Recommendations H.261 and H.263)
GSTN	General Switched Telephone Network
HDLC	High-level Data Link Control
HRD	Hypothetical Reference Decoder (refer to Recommendations H.261 and H.263)
IV	Initialization Vector (used for encryption: refer to Recommendations H.233 and H.234)
LAPM	Link Access Protocol for Modems
LCSE	Logical Channel Signalling Entity
MC	H.323 Multipoint Control Entity
MCU	Multipoint Control Unit
MLSE	Maintenance Loop Signalling Entity
MPI	Minimum Picture Interval
MRSE	Mode Request Signalling Entity
MSDSE	Master Slave Determination Signalling Entity
MTSE	Multiplex Table Signalling Entity
PCR	Program Clock Reference (refer to ITU-T Rec. H.222.0 ISO/IEC 13818-1)

PID	Packet Identifier (refer to ITU-T Rec. H.222.0 ISO/IEC 13818-1)
QCIF	Quarter CIF
RMESE	Request Multiplex Entry Signalling Entity
RTCP	Real-time Transport Control Protocol
RTDSE	Round Trip Delay Signalling Entity
RTP	Real-time Transport Protocol
SDL	Specification and Description Language
SDU	Service Data Unit
SE	Session Exchange Message (used for encryption: refer to Recommendations H.233 and H.234)
SQCIF	Sub QCIF
STD	System Target Decoder (refer to ITU-T Rec. H.222.0 ISO/IEC 13818-1)
VC	ATM Virtual Channel

5 General

This Recommendation provides a number of different services, some of which are expected to be applicable to all terminals that use it and some that are more specific to particular ones. Procedures are defined to allow the exchange of audiovisual and data capabilities, to request the transmission of a particular audiovisual and data mode, to manage the logical channels used to transport the audiovisual and data information, to establish which terminal is the master terminal and which is the slave terminal for the purposes of managing logical channels, to carry various control and indication signals, to control the bit rate of individual logical channels and the whole multiplex, and to measure the round trip delay, from one terminal to the other and back. These procedures are explained in more detail below.

Following this general introduction, there are clauses detailing the message syntax and semantics and the procedures. The syntax has been defined using ASN.1 notation [38] and the semantics define the meaning of syntax elements as well as providing syntactic constraints that are not specified in the ASN.1 syntax. The procedures clause defines the protocols that use the messages defined in the other clauses.

Although not all of the messages and procedures defined in this Recommendation will be applicable to all terminals, no indication of such restrictions is given here. These restrictions are the responsibility of the Recommendations that use this Recommendation.

This Recommendation has been defined to be independent of the underlying transport mechanism, but is intended to be used with a reliable transport layer, that is, one that provides guaranteed delivery of correct data.

5.1 Master-slave determination

Conflicts may arise when two terminals involved in a call initiate similar events simultaneously and only one such event is possible or desired, for example, when resources are available for only one occurrence of the event. To resolve such conflicts, one terminal shall act as a master and the other terminal shall act as a slave terminal. Rules specify how the master and slave terminal shall respond at times of conflict.

The master-slave determination procedure allows terminals in a call to determine which terminal is the master and which terminal is the slave. The terminal status may be re-determined at any time during a call; however, a terminal may only initiate the master-slave determination process if no procedure which depends upon its result is locally active.

5.2 Capability exchange

The capability exchange procedures are intended to ensure that the only multimedia signals to be transmitted are those that can be received and treated appropriately by the receive terminal. This requires that the capabilities of each terminal to receive and decode be known to the other terminal. It is not necessary that a terminal understand or store all incoming capabilities; those that are not understood, or cannot be used shall be ignored, and no fault shall be considered to have occurred. When a capability is received which contains extensions not understood by the terminal, the capability shall be accepted as if it did not contain the extensions.

The total capability of a terminal to receive and decode various signals is made known to the other terminal by transmission of its capability set.

Receive capabilities describe the terminal's ability to receive and process incoming information streams. Transmitters shall limit the content of their transmitted information to that which the receiver has indicated it is capable of receiving. The absence of a receive capability indicates that the terminal cannot receive (is a transmitter only).

Transmit capabilities describe the terminal's ability to transmit information streams. Transmit capabilities serve to offer receivers a choice of possible modes of operation, so that the receiver may request the mode which it prefers to receive. The absence of a transmit capability indicates that the terminal is not offering a choice of preferred modes to the receiver (but it may still transmit anything within the capability of the receiver).

These capability sets provide for more than one stream of a given medium type to be sent simultaneously. For example, a terminal may declare its ability to receive (or send) two independent H.262 video streams and two independent G.722 audio streams at the same time. Capability messages have been defined to allow a terminal to indicate that it does not have fixed capabilities, but that they depend on which other modes are being used simultaneously. For example, it is possible to indicate that higher resolution video can be decoded when a simpler audio algorithm is used; or that either two low resolution video sequences can be decoded or a single high resolution one. It is also possible to indicate trade-offs between the capability to transmit and the capability to receive.

Non-standard capabilities and control messages may be issued using the NonStandardParameter structure. Note that while the meaning of non-standard messages is defined by individual organizations, equipment built by any manufacturer may signal any non-standard message, if the meaning is known.

Terminals may re-issue capability sets at any time.

5.3 Logical channel signalling procedures

An acknowledged protocol is defined for the opening and closing of logical channels which carry the audiovisual and data information. The aim of these procedures is to ensure that a terminal is capable of receiving and decoding the data that will be transmitted on a logical channel at the time the logical channel is opened rather than at the time the first data is transmitted on it; and to ensure that the receive terminal is ready to receive and decode the data that will be transmitted on the logical channel before that transmission starts. The OpenLogicalChannel message includes a description of the data to be transported, for example, H.262 MP@ML at 6 Mbit/s. Logical channels should only be

opened when there is sufficient capability to receive data on all open logical channels simultaneously.

A part of this protocol is concerned with the opening of bidirectional channels. To avoid conflicts which may arise when two terminals initiate similar events simultaneously, one terminal is defined as the master terminal, and the other as the slave terminal. A protocol is defined to establish which terminal is the master and which is the slave. However, systems that use this Recommendation may specify the procedure specified in this Recommendation or another means of determining which terminal is the master and which is the slave.

5.4 Receive terminal close logical channel request

A logical channel is opened and closed from the transmitter side. A mechanism is defined which allows a receive terminal to request the closure of an incoming logical channel. The transmit terminal may accept or reject the logical channel closure request. A terminal may, for example, use these procedures to request the closure of an incoming logical channel which, for whatever reason, cannot be decoded. These procedures may also be used to request the closure of a bidirectional logical channel by the terminal that did not open the channel.

5.5 H.223 multiplex table entry modification

The H.223 multiplex table associates each octet within an H.223 MUX message with a particular logical channel number. The H.223 multiplex table may have up to 15 entries. A mechanism is provided that allows the transmit terminal to specify and inform the receiver of new H.223 multiplex table entries. A receive terminal may also request the retransmission of a multiplex table entry.

5.6 Audiovisual and data mode request

When the capability exchange protocol has been completed, both terminals will be aware of each other's capability to transmit and receive as specified in the capability descriptors that have been exchanged. It is not mandatory for a terminal to declare all its capabilities; it need only declare those that it wishes to be used.

A terminal may indicate its capabilities to transmit. A terminal that receives transmission capabilities from the remote terminal may request a particular mode to be transmitted to it. A terminal indicates that it does not want its transmission mode to be controlled by the remote terminal by sending no transmission capabilities.

5.7 Round-trip delay determination

It may be useful in some applications to have knowledge of the round-trip delay between a transmit terminal and a receive terminal. A mechanism is provided to measure this round-trip delay. This mechanism may also be useful as a means to detect whether the remote terminal is still functioning.

5.8 Maintenance loops

Procedures are specified to establish maintenance loops. It is possible to specify the loop of a single logical channel either as a digital loop or decoded loop, and the loop of the whole multiplex.

5.9 Commands and indications

Commands and indications are provided for various purposes: video/audio active/inactive signals to inform the user; fast update request for source switching in multipoint applications are some examples. Neither commands nor indications elicit response messages from the remote terminal.

Commands force an action at the remote terminal whilst indications merely provide information and do not force any action.

A command is defined to allow the bit rate of logical channels and the whole multiplex to be controlled from the remote terminal. This has a number of purposes: interworking with terminals using multiplexes in which only a finite number of bit rates are available; multipoint applications where the rates from different sources should be matched; and flow control in congested networks.

ANNEX A

Messages: Syntax

This annex specifies the syntax of messages using the notation defined in ASN.1 [38]. Messages shall be encoded for transmission by applying the packed encoding rules specified in [39] using the basic aligned variant. The first bit in each octet which is transmitted is the most significant bit of the octet as is specified in Recommendation X.691.

MULTIMEDIA-SYSTEM-CONTROL DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- Export all symbols

-- Top level Messages

```
MultimediaSystemControlMessage ::=CHOICE
{
    request      RequestMessage,
    response     ResponseMessage,
    command      CommandMessage,
    indication   IndicationMessage,
    ...
}
```

-- A RequestMessage results in action and requires an immediate response

```
RequestMessage ::=CHOICE
{
    nonStandard      NonStandardMessage,
    masterSlaveDetermination MasterSlaveDetermination,
    terminalCapabilitySet TerminalCapabilitySet,
    openLogicalChannel OpenLogicalChannel,
    closeLogicalChannel CloseLogicalChannel,
    requestChannelClose RequestChannelClose,
    multiplexEntrySend MultiplexEntrySend,
    requestMultiplexEntry RequestMultiplexEntry,
    requestMode      RequestMode,
    roundTripDelayRequest RoundTripDelayRequest,
```

maintenanceLoopRequest	MaintenanceLoopRequest,
....	
communicationModeRequest	CommunicationModeRequest,
conferenceRequest	ConferenceRequest,
multilinkRequest	MultilinkRequest
}	

-- A ResponseMessage is the response to a RequestMessage

ResponseMessage	::=CHOICE
{	
nonStandard	NonStandardMessage,
masterSlaveDeterminationAck	MasterSlaveDeterminationAck,
masterSlaveDeterminationReject	MasterSlaveDeterminationReject,
terminalCapabilitySetAck	TerminalCapabilitySetAck,
terminalCapabilitySetReject	TerminalCapabilitySetReject,
openLogicalChannelAck	OpenLogicalChannelAck,
openLogicalChannelReject	OpenLogicalChannelReject,
closeLogicalChannelAck	CloseLogicalChannelAck,
requestChannelCloseAck	RequestChannelCloseAck,
requestChannelCloseReject	RequestChannelCloseReject,
multiplexEntrySendAck	MultiplexEntrySendAck,
multiplexEntrySendReject	MultiplexEntrySendReject,
requestMultiplexEntryAck	RequestMultiplexEntryAck,
requestMultiplexEntryReject	RequestMultiplexEntryReject,
requestModeAck	RequestModeAck,
requestModeReject	RequestModeReject,
roundTripDelayResponse	RoundTripDelayResponse,
maintenanceLoopAck	MaintenanceLoopAck,
maintenanceLoopReject	MaintenanceLoopReject,
....	
communicationModeResponse	CommunicationModeResponse,
conferenceResponse	ConferenceResponse,
multilinkResponse	MultilinkResponse
}	

-- A CommandMessage requires action, but no explicit response

CommandMessage	::=CHOICE
{	
nonStandard	NonStandardMessage,
maintenanceLoopOffCommand	MaintenanceLoopOffCommand,
sendTerminalCapabilitySet	SendTerminalCapabilitySet,
encryptionCommand	EncryptionCommand,

flowControlCommand	FlowControlCommand,
endSessionCommand	EndSessionCommand,
miscellaneousCommand	MiscellaneousCommand,
....	
communicationModeCommand	CommunicationModeCommand,
conferenceCommand	ConferenceCommand,
h223MultiplexReconfiguration	H223MultiplexReconfiguration,
newATMVCCCommand	NewATMVCCCommand

-- *An IndicationMessage is information that does not require action or response*

IndicationMessage	::=CHOICE
{	
nonStandard	NonStandardMessage,
functionNotUnderstood	FunctionNotUnderstood,
masterSlaveDeterminationRelease	MasterSlaveDeterminationRelease,
terminalCapabilitySetRelease	TerminalCapabilitySetRelease,
openLogicalChannelConfirm	OpenLogicalChannelConfirm,
requestChannelCloseRelease	RequestChannelCloseRelease,
multiplexEntrySendRelease	MultiplexEntrySendRelease,
requestMultiplexEntryRelease	RequestMultiplexEntryRelease,
requestModeRelease	RequestModeRelease,
miscellaneousIndication	MiscellaneousIndication,
jitterIndication	JitterIndication,
h223SkewIndication	H223SkewIndication,
newATMVCIndication	NewATMVCIndication,
userInput	UserInputIndication,
....	
h2250MaximumSkewIndication	H2250MaximumSkewIndication,
mcLocationIndication	MCLocationIndication,
conferenceIndication	ConferenceIndication,
vendorIdentification	VendorIdentification,
functionNotSupported	FunctionNotSupported,
multilinkIndication	MultilinkIndication
}	

-- SequenceNumber is defined here as it is used in a number of Messages

SequenceNumber ::=INTEGER (0..255)

-- Non-standard Message definitions

NonStandardMessage ::=SEQUENCE
{
 nonStandardData NonStandardParameter,
 ...
}

NonStandardParameter ::=SEQUENCE
{
 nonStandardIdentifier NonStandardIdentifier,
 data OCTET STRING
}

NonStandardIdentifier ::=CHOICE
{
 object OBJECT IDENTIFIER,
 h221NonStandard SEQUENCE
 {
 t35CountryCode INTEGER (0..255), -- country, per T.35
 t35Extension INTEGER (0..255), -- assigned nationally
 manufacturerCode INTEGER (0..65535) -- assigned nationally
 }
}

-- Master-slave determination definitions

MasterSlaveDetermination ::=SEQUENCE
{
 terminalType INTEGER (0..255),
 statusDeterminationNumber INTEGER (0..16777215),
 ...
}

MasterSlaveDeterminationAck ::=SEQUENCE
{
 decision CHOICE
 {
 master NULL,
 slave NULL
 },
 ...
}

MasterSlaveDeterminationReject ::=SEQUENCE
{
 cause CHOICE
 {
 identicalNumbers NULL,
 ...
 },
 ...
}

```

MasterSlaveDeterminationRelease      ::=SEQUENCE
{
    ...
}

-----
-- Capability exchange definitions
-----

TerminalCapabilitySet                ::=SEQUENCE
{
    sequenceNumber                    SequenceNumber,

    protocolIdentifier                OBJECT IDENTIFIER,
                                        -- shall be set to the value
                                        -- {itu-t (0) recommendation (0) h (8) 245 version (0) 4}

    multiplexCapability                MultiplexCapability OPTIONAL,

    capabilityTable                    SET SIZE (1..256) OF CapabilityTableEntry OPTIONAL,

    capabilityDescriptors              SET SIZE (1..256) OF CapabilityDescriptor OPTIONAL,

    ...
}

CapabilityTableEntry                 ::=SEQUENCE
{
    capabilityTableEntryNumber         CapabilityTableEntryNumber,
    capability                         Capability OPTIONAL
}

CapabilityDescriptor                 ::=SEQUENCE
{
    capabilityDescriptorNumber         CapabilityDescriptorNumber,
    simultaneousCapabilities           SET SIZE (1..256) OF AlternativeCapabilitySet OPTIONAL
}

AlternativeCapabilitySet              ::=SEQUENCE SIZE (1..256) OF CapabilityTableEntryNumber

CapabilityTableEntryNumber           ::=INTEGER (1..65535)

CapabilityDescriptorNumber           ::=INTEGER (0..255)

TerminalCapabilitySetAck              ::=SEQUENCE
{
    sequenceNumber                    SequenceNumber,
    ...
}

TerminalCapabilitySetReject          ::=SEQUENCE
{
    sequenceNumber                    SequenceNumber,
    cause                             CHOICE
    {
        unspecified                   NULL,
        undefinedTableEntryUsed       NULL,
        descriptorCapacityExceeded    NULL,
    }
}

```



```

        tableEntryCapacityExceeded    CHOICE
        {
            highestEntryNumberProcessed  CapabilityTableEntryNumber,
            noneProcessed                NULL
        },
        ...
    },
    ...
}

TerminalCapabilitySetRelease          ::=SEQUENCE
{
    ...
}

=====
-- Capability exchange definitions: Top level capability description
=====

Capability                            ::=CHOICE
{
    nonStandard                        NonStandardParameter,

    receiveVideoCapability              VideoCapability,
    transmitVideoCapability             VideoCapability,
    receiveAndTransmitVideoCapability  VideoCapability,

    receiveAudioCapability              AudioCapability,
    transmitAudioCapability             AudioCapability,
    receiveAndTransmitAudioCapability  AudioCapability,

    receiveDataApplicationCapability    DataApplicationCapability,
    transmitDataApplicationCapability    DataApplicationCapability,
    receiveAndTransmitDataApplicationCapability  DataApplicationCapability,

    h233EncryptionTransmitCapability    BOOLEAN,
    h233EncryptionReceiveCapability     SEQUENCE
    {
        h233IVResponseTime             INTEGER (0..255),          -- units milliseconds
        ...
    },
    ...,
    conferenceCapability                ConferenceCapability,
    h235SecurityCapability               H235SecurityCapability,
    maxPendingReplacementFor            INTEGER (0..255),
    receiveUserInputCapability           UserInputCapability,
    transmitUserInputCapability          UserInputCapability,
    receiveAndTransmitUserInputCapability  UserInputCapability
}

H235SecurityCapability                ::=SEQUENCE
{
    encryptionAuthenticationAndIntegrity  EncryptionAuthenticationAndIntegrity,

    mediaCapability                      CapabilityTableEntryNumber,
    -- Note: The mediaCapability shall refer to Capability Table Entries that do contain a transmit,
    -- receive, or receiveAndTransmit AudioCapability, VideoCapability,
    -- DataApplicationCapability, or similar capability indicated by a NonStandardParameter only
    ...
}

```

```

-----
-- Capability exchange definitions: Multiplex capabilities
-----

```

```

MultiplexCapability ::=CHOICE
{
    nonStandard          NonStandardParameter,
    h222Capability       H222Capability,
    h223Capability       H223Capability,
    v76Capability        V76Capability,
    ...,
    h2250Capability      H2250Capability
}

H222Capability ::= SEQUENCE
{
    numberOfVCs          INTEGER (1..256),
    vcCapability          SET OF VCCapability,
    ...
}

VCCapability ::=SEQUENCE
{
    aal1      SEQUENCE
    {
        nullClockRecovery      BOOLEAN,
        srtsClockRecovery      BOOLEAN,
        adaptiveClockRecovery  BOOLEAN,
        nullErrorCorrection    BOOLEAN,
        longInterleaver        BOOLEAN,
        shortInterleaver       BOOLEAN,
        errorCorrectionOnly    BOOLEAN,
        structuredDataTransfer  BOOLEAN,
        partiallyFilledCells   BOOLEAN,
        ...
    } OPTIONAL,
    aal5      SEQUENCE
    {
        forwardMaximumSDUSize  INTEGER (0..65535),      -- units octets
        backwardMaximumSDUSize INTEGER (0..65535),      -- units octets
        ...
    } OPTIONAL,
    transportStream            BOOLEAN,
    programStream              BOOLEAN,
    availableBitRates          SEQUENCE
    {
        type                    CHOICE
        {
            singleBitRate       INTEGER (1..65535),      -- units 64 kbit/s
            rangeOfBitRates     SEQUENCE
            {
                lowerBitRate    INTEGER (1..65535),      -- units 64 kbit/s
                higherBitRate   INTEGER (1..65535),      -- units 64 kbit/s
            }
        },
        ...
    },
    ...
}

```

aal1ViaGateway	SEQUENCE	
{		
gatewayAddress	SET SIZE(1...256) OF Q2931Address,	
nullClockRecovery	BOOLEAN,	
srtsClockRecovery	BOOLEAN,	
adaptiveClockRecovery	BOOLEAN,	
nullErrorCorrection	BOOLEAN,	
longInterleaver	BOOLEAN,	
shortInterleaver	BOOLEAN,	
errorCorrectionOnly	BOOLEAN,	
structuredDataTransfer	BOOLEAN,	
partiallyFilledCells	BOOLEAN,	
...		
} OPTIONAL		
}		
H223Capability	::=SEQUENCE	
{		
transportWithI-frames	BOOLEAN,	<i>-- I-frame transport of H.245</i>
videoWithAL1	BOOLEAN,	
videoWithAL2	BOOLEAN,	
videoWithAL3	BOOLEAN,	
audioWithAL1	BOOLEAN,	
audioWithAL2	BOOLEAN,	
audioWithAL3	BOOLEAN,	
dataWithAL1	BOOLEAN,	
dataWithAL2	BOOLEAN,	
dataWithAL3	BOOLEAN,	
maximumA12SDUSize	INTEGER (0..65535),	<i>-- units octets</i>
maximumA13SDUSize	INTEGER (0..65535),	<i>-- units octets</i>
maximumDelayJitter	INTEGER (0..1023),	<i>-- units milliseconds</i>
h223MultiplexTableCapability	CHOICE	
{		
basic	NULL,	
enhanced	SEQUENCE	
{		
maximumNestingDepth	INTEGER (1..15),	
maximumElementListSize	INTEGER (2..255),	
maximumSubElementListSize	INTEGER (2..255),	
...		
}		
},		
...,		
maxMUXPDUSizeCapability	BOOLEAN,	
nsrpSupport	BOOLEAN,	
mobileOperationTransmitCapability	SEQUENCE	
{		
modeChangeCapability	BOOLEAN,	
h223AnnexA	BOOLEAN,	
h223AnnexADoubleFlag	BOOLEAN,	
h223AnnexB	BOOLEAN,	
h223AnnexBwithHeader	BOOLEAN,	
...		
} OPTIONAL,		
h223AnnexCCapability	H223AnnexCCapability OPTIONAL	
}		

```

H223AnnexCCapability ::= SEQUENCE
{
    videoWithAL1M          BOOLEAN,
    videoWithAL2M          BOOLEAN,
    videoWithAL3M          BOOLEAN,
    audioWithAL1M          BOOLEAN,
    audioWithAL2M          BOOLEAN,
    audioWithAL3M          BOOLEAN,
    dataWithAL1M           BOOLEAN,
    dataWithAL2M           BOOLEAN,
    dataWithAL3M           BOOLEAN,
    alpduInterleaving      BOOLEAN,

    maximumAL1MPDUSize     INTEGER (0..65535),    -- units octets
    maximumAL2MSDUSize     INTEGER (0..65535),    -- units octets
    maximumAL3MSDUSize     INTEGER (0..65535),    -- units octets
    ...
}

V76Capability ::=SEQUENCE
{
    suspendResumeCapabilitywAddress    BOOLEAN,
    suspendResumeCapabilitywoAddress   BOOLEAN,
    rejCapability                       BOOLEAN,
    sREJCapability                      BOOLEAN,
    mREJCapability                      BOOLEAN,
    crc8bitCapability                   BOOLEAN,
    crc16bitCapability                  BOOLEAN,
    crc32bitCapability                  BOOLEAN,
    uihCapability                       BOOLEAN,
    numOfDLCS                           INTEGER (2..8191),
    twoOctetAddressFieldCapability     BOOLEAN,
    loopBackTestCapability              BOOLEAN,
    n401Capability                      INTEGER (1..4095),
    maxWindowSizeCapability             INTEGER (1..127),
    v75Capability                       V75Capability,
    ...
}

V75Capability ::=SEQUENCE
{
    audioHeader                    BOOLEAN,
    ...
}

H2250Capability ::=SEQUENCE
{
    maximumAudioDelayJitter          INTEGER(0..1023),    -- units in milliseconds
    receiveMultipointCapability       MultipointCapability,
    transmitMultipointCapability      MultipointCapability,
    receiveAndTransmitMultipointCapability MultipointCapability,
    mcCapability                      SEQUENCE
    {
        centralizedConferenceMC       BOOLEAN,
        decentralizedConferenceMC     BOOLEAN,
        ...
    },
    rtcpVideoControlCapability        BOOLEAN,    -- FIR and NACK
    mediaPacketizationCapability      MediaPacketizationCapability,
    ...,
}

```

```

    transportCapability          TransportCapability OPTIONAL,
    redundancyEncodingCapability SEQUENCE SIZE(1..256) OF
                                RedundancyEncodingCapability OPTIONAL,
                                BOOLEAN,
                                BOOLEAN
}

MediaPacketizationCapability ::=SEQUENCE
{
    h261aVideoPacketization    BOOLEAN,
    ...,
    rtpPayloadType             SEQUENCE SIZE(1..256) OF RTPPayloadType
                                OPTIONAL
}

RSVPPParameters ::=SEQUENCE
{
    qosMode                    QOSMode OPTIONAL,
    tokenRate                  INTEGER (1..4294967295) OPTIONAL,
                                -- rate in bytes/sec
    bucketSize                  INTEGER (1..4294967295) OPTIONAL,
                                -- size in bytes
    peakRate                   INTEGER (1..4294967295) OPTIONAL,
                                -- peak bandwidth bytes/sec
    minPoliced                  INTEGER (1..4294967295) OPTIONAL,
    maxPktSize                 INTEGER (1..4294967295) OPTIONAL,
                                -- size in bytes
    ...
}

QOSMode ::=CHOICE
{
    guaranteedQOS              NULL,
    controlledLoad              NULL,
    ...
}

-- Ed. Replace with ITU-T ATM Transfer Capability Types
ATMPParameters ::=SEQUENCE
{
    maxNTUSize                 INTEGER(0..65535),
                                -- units in octets
    atmUBR                     BOOLEAN,
                                -- unspecified bit rate
    atmrtVBR                   BOOLEAN,
                                -- real time variable bit rate
    atmrrtVBR                  BOOLEAN,
                                -- non-real time variable bit rate
    atmABR                     BOOLEAN,
                                -- available bit rate
    atmCBR                     BOOLEAN,
                                -- constant bit rate
    ...
}

QOSCapability ::=SEQUENCE
{
    nonStandardData            NonStandardParameter OPTIONAL,
    rsvpParameters             RSVPPParameters OPTIONAL,
    atmParameters              ATMPParameters OPTIONAL,
    ...
}

MediaTransportType ::=CHOICE
{
    ip-UDP                     NULL,
    ip-TCP                     NULL,
    atm-AAL5-UNIDIR            NULL, -- virtual circuits used as unidirectional
}

```

<pre> atm-AAL5-BIDIR ... } MediaChannelCapability { mediaTransport ... } TransportCapability { nonStandard qosCapabilities mediaChannelCapabilities ... } RedundancyEncodingCapability { redundancyEncodingMethod primaryEncoding secondaryEncoding ... } RedundancyEncodingMethod { nonStandard rtpAudioRedundancyEncoding ..., rtpH263VideoRedundancyEncoding } RTPH263VideoRedundancyEncoding { numberOfThreads framesBetweenSyncPoints frameToThreadMapping { roundrobin custom ... }, containedThreads ... } RTPH263VideoRedundancyFrameMapping { threadNumber frameSequence ... } </pre>	<pre> NULL, -- virtual circuits used as bidirectional ::=SEQUENCE MediaTransportType OPTIONAL, ::=SEQUENCE NonStandardParameter OPTIONAL, SEQUENCE SIZE(1..256) OF QOSCapability OPTIONAL, SEQUENCE SIZE(1..256) OF MediaChannelCapability OPTIONAL, ::=SEQUENCE RedundancyEncodingMethod, CapabilityTableEntryNumber, SEQUENCE SIZE(1..256) OF CapabilityTableEntryNumber OPTIONAL, ::=CHOICE NonStandardParameter, NULL, RTPH263VideoRedundancyEncoding ::= SEQUENCE INTEGER (1..16), INTEGER (1..256), CHOICE NULL, SEQUENCE SIZE(1..256) OF RTPH263VideoRedundancyFrameMapping, -- empty SEQUENCE for capability negotiation -- meaningful contents only OpenLogicalChannel SEQUENCE SIZE(1..256) OF INTEGER (0..15) OPTIONAL, -- only used for opening of logical channels ::= SEQUENCE INTEGER (0..15), SEQUENCE SIZE(1..256) OF INTEGER (0..255), </pre>
--	---

```

MultipointCapability ::=SEQUENCE
{
    multicastCapability          BOOLEAN,
    multiUniCastConference      BOOLEAN,
    mediaDistributionCapability SEQUENCE OF MediaDistributionCapability,
    ...
}

MediaDistributionCapability ::=SEQUENCE
{
    centralizedControl          BOOLEAN,
    distributedControl          BOOLEAN,
    centralizedAudio           BOOLEAN,
    distributedAudio          BOOLEAN,
    centralizedVideo          BOOLEAN,
    distributedVideo          BOOLEAN,
    centralizedData           SEQUENCE OF DataApplicationCapability OPTIONAL,
    distributedData           SEQUENCE OF DataApplicationCapability OPTIONAL,
    -- for further study in H.323
    ...
}

```

```

-- =====
-- Capability exchange definitions: Video capabilities
-- =====

```

```

VideoCapability ::=CHOICE
{
    nonStandard                NonStandardParameter,
    h261VideoCapability        H261VideoCapability,
    h262VideoCapability        H262VideoCapability,
    h263VideoCapability        H263VideoCapability,
    is11172VideoCapability     IS11172VideoCapability,
    ...
}

H261VideoCapability ::=SEQUENCE
{
    qcifMPI                    INTEGER (1..4) OPTIONAL, -- units 1/29.97 Hz
    cifMPI                     INTEGER (1..4) OPTIONAL, -- units 1/29.97 Hz
    temporalSpatialTradeOffCapability BOOLEAN,
    maxBitRate                 INTEGER (1..19200), -- units of 100 bit/s
    stillImageTransmission     BOOLEAN, -- Annex D/H.261
    ...
}

H262VideoCapability ::=SEQUENCE
{
    profileAndLevel-SPatML      BOOLEAN,
    profileAndLevel-MPatLL      BOOLEAN,
    profileAndLevel-MPatML      BOOLEAN,
    profileAndLevel-MPatH-14    BOOLEAN,
    profileAndLevel-MPatHL      BOOLEAN,
    profileAndLevel-SNRatLL     BOOLEAN,
    profileAndLevel-SNRatML     BOOLEAN,
    profileAndLevel-SpatialatH-14 BOOLEAN,
    profileAndLevel-HPatML      BOOLEAN,
    profileAndLevel-HPatH-14    BOOLEAN,

```

```

profileAndLevel-HPatHL
videoBitRate          BOOLEAN,
vbvBufferSize         INTEGER (0.. 1073741823) OPTIONAL, -- units 400 bit/s
samplesPerLine        INTEGER (0.. 262143) OPTIONAL,   -- units 16 384 bits
linesPerFrame         INTEGER (0..16383) OPTIONAL,   -- units samples/line
framesPerSecond       INTEGER (0..16383) OPTIONAL,   -- units lines/frame
luminanceSampleRate   INTEGER (0..15) OPTIONAL,      -- frame_rate_code
...
}

H263VideoCapability ::=SEQUENCE
{
    sqcifMPI          INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    qcifMPI           INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    cifMPI            INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    cif4MPI           INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    cif16MPI          INTEGER (1..32) OPTIONAL,      -- units 1/29.97 Hz
    maxBitRate        INTEGER (1..192400),           -- units 100 bit/s
    unrestrictedVector BOOLEAN,
    arithmeticCoding  BOOLEAN,
    advancedPrediction BOOLEAN,
    pbFrames           BOOLEAN,
    temporalSpatialTradeOffCapability BOOLEAN,
    hrd-B              INTEGER (0..524287) OPTIONAL, -- units 128 bits
    bppMaxKb           INTEGER (0..65535) OPTIONAL,  -- units 1024 bits
    ...,

    slowSqcifMPI      INTEGER (1..3600) OPTIONAL, -- units seconds/frame
    slowQcifMPI        INTEGER (1..3600) OPTIONAL, -- units seconds/frame
    slowCifMPI         INTEGER (1..3600) OPTIONAL, -- units seconds/frame
    slowCif4MPI        INTEGER (1..3600) OPTIONAL, -- units seconds/frame
    slowCif16MPI       INTEGER (1..3600) OPTIONAL, -- units seconds/frame
    errorCompensation  BOOLEAN,

    enhancementLayerInfo EnhancementLayerInfo OPTIONAL,
    h263Options         H263Options OPTIONAL
}

EnhancementLayerInfo ::=SEQUENCE
{
    baseBitRateConstrained BOOLEAN,
    snrEnhancement         SET SIZE(1..14) OF EnhancementOptions OPTIONAL,
    spatialEnhancement     SET SIZE(1..14) OF EnhancementOptions OPTIONAL,
    bPictureEnhancement    SET SIZE(1..14) OF EnhancementParameters OPTIONAL,
    ...
}

BEnhancementParameters ::=SEQUENCE
{
    enhancementOptions EnhancementOptions,
    numberOfBPictures  INTEGER (1..64),
    ...
}

EnhancementOptions ::=SEQUENCE
{
    sqcifMPI          INTEGER (1..32) OPTIONAL, -- units 1/29.97 Hz
    qcifMPI           INTEGER (1..32) OPTIONAL, -- units 1/29.97 Hz
    cifMPI            INTEGER (1..32) OPTIONAL, -- units 1/29.97 Hz
    cif4MPI           INTEGER (1..32) OPTIONAL, -- units 1/29.97 Hz

```


cif16MPI	INTEGER (1..32) OPTIONAL, -- units 1/29.97 Hz
maxBitRate	INTEGER (1..192400), -- units 100 bit/s
unrestrictedVector	BOOLEAN,
arithmeticCoding	BOOLEAN,
temporalSpatialTradeOffCapability	BOOLEAN,
slowSqcifMPI	INTEGER (1..3600) OPTIONAL, -- units seconds/frame
slowQcifMPI	INTEGER (1..3600) OPTIONAL, -- units seconds/frame
slowCifMPI	INTEGER (1..3600) OPTIONAL, -- units seconds/frame
slowCif4MPI	INTEGER (1..3600) OPTIONAL, -- units seconds/frame
slowCif16MPI	INTEGER (1..3600) OPTIONAL, -- units seconds/frame
errorCompensation	BOOLEAN,
h263Options	H263Options OPTIONAL,
...	
}	
H263Options	::= SEQUENCE
{	
advancedIntraCodingMode	BOOLEAN,
deblockingFilterMode	BOOLEAN,
improvedPBFramesMode	BOOLEAN,
unlimitedMotionVectors	BOOLEAN,
fullPictureFreeze	BOOLEAN,
partialPictureFreezeAndRelease	BOOLEAN,
resizingPartPicFreezeAndRelease	BOOLEAN,
fullPictureSnapshot	BOOLEAN,
partialPictureSnapshot	BOOLEAN,
videoSegmentTagging	BOOLEAN,
progressiveRefinement	BOOLEAN,
dynamicPictureResizingByFour	BOOLEAN,
dynamicPictureResizingSixteenthPel	BOOLEAN,
dynamicWarpingHalfPel	BOOLEAN,
dynamicWarpingSixteenthPel	BOOLEAN,
independentSegmentDecoding	BOOLEAN,
slicesInOrder-NonRect	BOOLEAN,
slicesInOrder-Rect	BOOLEAN,
slicesNoOrder-NonRect	BOOLEAN,
slicesNoOrder-Rect	BOOLEAN,
alternateInterVLCMode	BOOLEAN,
modifiedQuantizationMode	BOOLEAN,
reducedResolutionUpdate	BOOLEAN,
transparencyParameters	TransparencyParameters OPTIONAL,
separateVideoBackChannel	BOOLEAN,
refPictureSelection	RefPictureSelection OPTIONAL,
customPictureClockFrequency	SET SIZE (1..16) OF CustomPictureClockFrequency
	OPTIONAL,
customPictureFormat	SET SIZE (1..16) OF CustomPictureFormat OPTIONAL,
modeCombos	SET SIZE (1..16) OF H263VideoModeCombos OPTIONAL,
...	
}	
TransparencyParameters	::= SEQUENCE
{	
presentationOrder	INTEGER(1..256),
offset-x	INTEGER(-262144..262143), -- 1/8 pixels

<pre> offset-y scale-x scale-y ... } </pre>	<pre> INTEGER(-262144..262143), -- 1/8 pixels INTEGER(1..255), INTEGER(1..255), </pre>
<pre> RefPictureSelection { additionalPictureMemory { sqcifAdditionalPictureMemory qcifAdditionalPictureMemory cifAdditionalPictureMemory cif4AdditionalPictureMemory cif16AdditionalPictureMemory bigCpfAdditionalPictureMemory ... } OPTIONAL, videoMux videoBackChannelSend { none ackMessageOnly nackMessageOnly ackOrNackMessageOnly ackAndNackMessage ... }, ... } </pre>	<pre> ::=SEQUENCE SEQUENCE INTEGER (1..256) OPTIONAL, -- units frame INTEGER (1..256) OPTIONAL, -- units frame INTEGER (1..256) OPTIONAL, -- units frame INTEGER (1..256) OPTIONAL, -- units frame INTEGER (1..256) OPTIONAL, -- units frame INTEGER (1..256) OPTIONAL, -- units frame BOOLEAN, CHOICE NULL, NULL, NULL, NULL, NULL, NULL, </pre>
<pre> CustomPictureClockFrequency { clockConversionCode clockDivisor sqcifMPI qcifMPI cifMPI cif4MPI cif16MPI ... } </pre>	<pre> ::=SEQUENCE INTEGER(1000..1001), INTEGER(1..127), INTEGER (1..2048) OPTIONAL, INTEGER (1..2048) OPTIONAL, INTEGER (1..2048) OPTIONAL, INTEGER (1..2048) OPTIONAL, INTEGER (1..2048) OPTIONAL, </pre>
<pre> CustomPictureFormat { maxCustomPictureWidth maxCustomPictureHeight minCustomPictureWidth minCustomPictureHeight mPI { standardMPI customPCF { clockConversionCode clockDivisor customMPI ... } OPTIONAL, ... }, } </pre>	<pre> ::=SEQUENCE INTEGER(1..2048), -- units 4 pixels INTEGER(1..2048), -- units 4 pixels INTEGER(1..2048), -- units 4 pixels INTEGER(1..2048), -- units 4 pixels SEQUENCE INTEGER (1..31) OPTIONAL, SET SIZE (1..16) OF SEQUENCE INTEGER (1000..1001), INTEGER (1..127), INTEGER (1..2048), </pre>

<pre> pixelAspectInformation { anyPixelAspectRatio pixelAspectCode extendedPAR { width height ... }, ... } </pre>	<pre> CHOICE BOOLEAN, SET SIZE (1..14) OF INTEGER(1..14), SET SIZE (1..256) OF SEQUENCE INTEGER(1..255), INTEGER(1..255), </pre>
<pre> H263VideoModeCombos { h263VideoUncoupledModes h263VideoCoupledModes ... } </pre>	<pre> ::= SEQUENCE H263ModeComboFlags, SET SIZE (1..16) OF H263ModeComboFlags, </pre>
<pre> H263ModeComboFlags { unrestrictedVector arithmeticCoding advancedPrediction pbFrames advancedIntraCodingMode deblockingFilterMode unlimitedMotionVectors slicesInOrder-NonRect slicesInOrder-Rect slicesNoOrder-NonRect slicesNoOrder-Rect improvedPBFramesMode referencePicSelect dynamicPictureResizingByFour dynamicPictureResizingSixteenthPel dynamicWarpingHalfPel dynamicWarpingSixteenthPel reducedResolutionUpdate independentSegmentDecoding alternateInterVLCMode modifiedQuantizationMode ... } </pre>	<pre> ::= SEQUENCE BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, BOOLEAN, </pre>
<pre> IS11172VideoCapability { constrainedBitstream videoBitRate vbvBufferSize samplesPerLine linesPerFrame pictureRate luminanceSampleRate ... } </pre>	<pre> ::=SEQUENCE BOOLEAN, INTEGER (0.. 1073741823) OPTIONAL, -- units 400 bit/s INTEGER (0.. 262143) OPTIONAL, -- units 16 384 bits INTEGER (0..16383) OPTIONAL, -- units samples/line INTEGER (0..16383) OPTIONAL, -- units lines/frame INTEGER (0..15) OPTIONAL, INTEGER (0..4294967295) OPTIONAL, -- units samples/sec </pre>

=====
-- Capability exchange definitions: Audio capabilities
=====

-- For an H.222 multiplex, the integers indicate the size of the STD buffer in units of 256 octets
-- For an H.223 multiplex, the integers indicate the maximum number of audio frames per AL-SDU
-- For an H.225.0 multiplex, the integers indicate the maximum number of audio frames per packet

```
AudioCapability ::= CHOICE
{
    nonStandard NonStandardParameter,
    g711Alaw64k INTEGER (1..256),
    g711Alaw56k INTEGER (1..256),
    g711Ulaw64k INTEGER (1..256),
    g711Ulaw56k INTEGER (1..256),

    g722-64k INTEGER (1..256),
    g722-56k INTEGER (1..256),
    g722-48k INTEGER (1..256),

    g7231 SEQUENCE
    {
        maxAl-sduAudioFrames INTEGER (1..256),
        silenceSuppression BOOLEAN
    },

    g728 INTEGER (1..256),
    g729 INTEGER (1..256),
    g729AnnexA INTEGER (1..256),
    is11172AudioCapability IS11172AudioCapability,
    is13818AudioCapability IS13818AudioCapability,

    ...
    g729wAnnexB INTEGER(1..256),
    g729AnnexAwAnnexB INTEGER(1..256),
    g7231AnnexCCapability G7231AnnexCCapability,
    gsmFullRate GSMAudioCapability,
    gsmHalfRate GSMAudioCapability,
    gsmEnhancedFullRate GSMAudioCapability
}

G7231AnnexCCapability ::= SEQUENCE
{
    maxAl-sduAudioFrames INTEGER (1..256),
    silenceSuppression BOOLEAN,
    g723AnnexCAudioMode SEQUENCE
    {
        highRateMode0 INTEGER (27..78), -- units octets
        highRateMode1 INTEGER (27..78), -- units octets
        lowRateMode0 INTEGER (23..66), -- units octets
        lowRateMode1 INTEGER (23..66), -- units octets
        sidMode0 INTEGER (6..17), -- units octets
        sidMode1 INTEGER (6..17), -- units octets
        ...
    } OPTIONAL,
    ...
}
```

```

IS11172AudioCapability ::=SEQUENCE
{
    audioLayer1          BOOLEAN,
    audioLayer2          BOOLEAN,
    audioLayer3          BOOLEAN,

    audioSampling32k     BOOLEAN,
    audioSampling44k1   BOOLEAN,
    audioSampling48k     BOOLEAN,

    singleChannel        BOOLEAN,
    twoChannels          BOOLEAN,

    bitRate              INTEGER (1..448),      -- units kbit/s
    ...
}

```

```

IS13818AudioCapability ::=SEQUENCE
{
    audioLayer1          BOOLEAN,
    audioLayer2          BOOLEAN,
    audioLayer3          BOOLEAN,

    audioSampling16k     BOOLEAN,
    audioSampling22k05  BOOLEAN,
    audioSampling24k     BOOLEAN,
    audioSampling32k     BOOLEAN,
    audioSampling44k1   BOOLEAN,
    audioSampling48k     BOOLEAN,

    singleChannel        BOOLEAN,
    twoChannels          BOOLEAN,
    threeChannels2-1     BOOLEAN,
    threeChannels3-0     BOOLEAN,
    fourChannels2-0-2-0  BOOLEAN,
    fourChannels2-2     BOOLEAN,
    fourChannels3-1     BOOLEAN,
    fiveChannels3-0-2-0  BOOLEAN,
    fiveChannels3-2     BOOLEAN,

    lowFrequencyEnhancement  BOOLEAN,

    multilingual         BOOLEAN,

    bitRate              INTEGER (1..1130),    -- units kbit/s
    ...
}

```

```

GSMAudioCapability ::= SEQUENCE
{
    audioUnitSize        INTEGER (1..256),
    comfortNoise         BOOLEAN,
    scrambled             BOOLEAN,
    ...
}

```

 -- Capability exchange definitions: Data capabilities

```

DataApplicationCapability ::=SEQUENCE
{
  application CHOICE
  {
    nonStandard NonStandardParameter,
    t120 DataProtocolCapability,
    dsm-cc DataProtocolCapability,
    userData DataProtocolCapability,
    t84 SEQUENCE
    {
      t84Protocol DataProtocolCapability,
      t84Profile T84Profile
    },
    t434 DataProtocolCapability,
    h224 DataProtocolCapability,
    nlpid SEQUENCE
    {
      nlpidProtocol DataProtocolCapability,
      nlpidData OCTET STRING
    },
    dsvdControl NULL,
    h222DataPartitioning DataProtocolCapability,
    ...,
    t30fax DataProtocolCapability,
    t140 DataProtocolCapability,
    t38fax SEQUENCE
    {
      t38FaxProtocol DataProtocolCapability,
      t38FaxProfile T38FaxProfile
    }
  },
  maxBitRate INTEGER (0..4294967295), -- units 100 bit/s
  ...
}

DataProtocolCapability ::=CHOICE
{
  nonStandard NonStandardParameter,
  v14buffered NULL,
  v42lapm NULL, -- may negotiate to V.42 bis
  hdlcFrameTunnelling NULL,
  h310SeparateVCStack NULL,
  h310SingleVCStack NULL,
  transparent NULL,
  ...,
  segmentationAndReassembly NULL,
  hdlcFrameTunnelingwSAR NULL,
  v120 NULL, -- as in H.230
  separateLANStack NULL,
  v76wCompression CHOICE
  {
    transmitCompression CompressionType,
    receiveCompression CompressionType,
    transmitAndReceiveCompression CompressionType,
    ...
  },
}

```

```

    tcp                NULL,
    udp                NULL
}

CompressionType      ::=CHOICE
{
    v42bis            V42bis,
    ...
}

V42bis               ::=SEQUENCE
{
    numberOfCodewords    INTEGER (1..65536),
    maximumStringLength  INTEGER (1..256),
    ...
}

T84Profile           ::=CHOICE
{
    t84Unrestricted     NULL,
    t84Restricted        SEQUENCE
    {
        qcif            BOOLEAN,
        cif              BOOLEAN,
        ccir601Seq      BOOLEAN,
        ccir601Prog     BOOLEAN,
        hdtvSeq         BOOLEAN,
        hdtvProg        BOOLEAN,

        g3FacsMH200x100  BOOLEAN,
        g3FacsMH200x200  BOOLEAN,
        g4FacsMMR200x100  BOOLEAN,
        g4FacsMMR200x200  BOOLEAN,
        jbig200x200Seq   BOOLEAN,
        jbig200x200Prog  BOOLEAN,
        jbig300x300Seq   BOOLEAN,
        jbig300x300Prog  BOOLEAN,

        digPhotoLow     BOOLEAN,
        digPhotoMedSeq  BOOLEAN,
        digPhotoMedProg  BOOLEAN,
        digPhotoHighSeq  BOOLEAN,
        digPhotoHighProg  BOOLEAN,

        ...
    }
}

T38FaxProfile        ::=SEQUENCE
{
    fillBitRemoval      BOOLEAN,
    transcodingJBIG     BOOLEAN,
    transcodingMMR      BOOLEAN,
    ...
}

```

=====
-- Encryption Capability Definitions
=====

EncryptionAuthenticationAndIntegrity ::=SEQUENCE
{
 encryptionCapability EncryptionCapability OPTIONAL,
 authenticationCapability AuthenticationCapability OPTIONAL,
 integrityCapability IntegrityCapability OPTIONAL,
 ...
}

EncryptionCapability ::=SEQUENCE SIZE(1..256) OF
MediaEncryptionAlgorithm

MediaEncryptionAlgorithm ::=CHOICE
{
 nonStandard NonStandardParameter,
 algorithm OBJECT IDENTIFIER, -- many defined in ISO/IEC 9979
 ...
}

AuthenticationCapability ::=SEQUENCE
{
 nonStandard NonStandardParameter OPTIONAL,
 ...
}

IntegrityCapability ::=SEQUENCE
{
 nonStandard NonStandardParameter OPTIONAL,
 ...
}

=====
-- Capability Exchange Definitions: UserInput
=====

UserInputCapability ::= CHOICE
{
 nonStandard SEQUENCE SIZE(1..16) OF NonStandardParameter,
 basicString NULL, -- alphanumeric
 iA5String NULL, -- alphanumeric
 generalString NULL, -- alphanumeric
 dtmf NULL, -- supports dtmf using signal and signalUpdate
 hookflash NULL, -- supports hookflash using signal
 ...
}

=====
-- Capability Exchange Definitions: Conference
=====

ConferenceCapability ::=SEQUENCE
{
 nonStandardData SEQUENCE OF NonStandardParameter OPTIONAL,
 chairControlCapability BOOLEAN,
 ...
}


```

=====
-- Logical channel signalling definitions
=====

-- 'Forward' is used to refer to transmission in the direction from the terminal making the
-- original request for a logical channel to the other terminal, and 'reverse' is used to refer
-- to the opposite direction of transmission, in the case of a bidirectional channel request.

OpenLogicalChannel ::=SEQUENCE
{
    forwardLogicalChannelNumber LogicalChannelNumber,

    forwardLogicalChannelParameters SEQUENCE
    {
        portNumber INTEGER (0..65535) OPTIONAL,
        dataType DataType,
        multiplexParameters CHOICE
        {
            h222LogicalChannelParameters H222LogicalChannelParameters,
            h223LogicalChannelParameters H223LogicalChannelParameters,
            v76LogicalChannelParameters V76LogicalChannelParameters,
            ...,
            h2250LogicalChannelParameters H2250LogicalChannelParameters,
            none NULL -- for use with Separate Stack when
                -- multiplexParameters are not required
                -- or appropriate
        },
        ...,
        forwardLogicalChannelDependency LogicalChannelNumber OPTIONAL,
            -- also used to refer to the primary logical channel when using video redundancy coding
        replacementFor LogicalChannelNumber OPTIONAL
    },

    -- Used to specify the reverse channel for bidirectional open request

    reverseLogicalChannelParameters SEQUENCE
    {
        dataType DataType,
        multiplexParameters CHOICE
        {
            -- H.222 parameters are never present in reverse direction
            h223LogicalChannelParameters H223LogicalChannelParameters,
            v76LogicalChannelParameters V76LogicalChannelParameters,
            ...,
            h2250LogicalChannelParameters H2250LogicalChannelParameters
        } OPTIONAL, -- Not present for H.222
        ...,
        reverseLogicalChannelDependency LogicalChannelNumber OPTIONAL,
            -- also used to refer to the primary logical channel when using video redundancy coding
        replacementFor LogicalChannelNumber OPTIONAL
    } OPTIONAL, -- Not present for unidirectional channel request
    ...,
    separateStack NetworkAccessParameters OPTIONAL,
        -- for Open responder to establish the stack
    encryptionSync EncryptionSync OPTIONAL -- used only by Master
}

```

LogicalChannelNumber	::=INTEGER (1..65535)
NetworkAccessParameters	::=SEQUENCE
{	
distribution	CHOICE
{	
unicast	NULL,
multicast	NULL,
...	
} OPTIONAL,	
networkAddress	CHOICE
{	
q2931Address	Q2931Address,
e164Address	IA5String(SIZE(1..128)) (FROM ("0123456789#*,")),
localAreaAddress	TransportAddress,
...	
},	
associateConference	BOOLEAN,
externalReference	OCTET STRING(SIZE(1..255)) OPTIONAL,
...,	
t120SetupProcedure	CHOICE
{	
originateCall	NULL,
waitForCall	NULL,
issueQuery	NULL,
...	
} OPTIONAL	
}	
Q2931Address	::=SEQUENCE
{	
address	CHOICE
{	
internationalNumber	NumericString(SIZE(1..16)),
nsapAddress	OCTET STRING (SIZE(1..20)),
...	
},	
subaddress	OCTET STRING (SIZE(1..20)) OPTIONAL,
...	
}	
V75Parameters	::= SEQUENCE
{	
audioHeaderPresent	BOOLEAN,
...	
}	
DataType	::=CHOICE
{	
nonStandard	NonStandardParameter,
nullData	NULL,
videoData	VideoCapability,
audioData	AudioCapability,
data	DataApplicationCapability,
encryptionData	EncryptionMode,
...,	

<pre> h235Control h235Media } </pre>	<pre> NonStandardParameter, H235Media </pre>
<pre> H235Media { encryptionAuthenticationAndIntegrity mediaType { nonStandard videoData audioData data ... }, ... } </pre>	<pre> ::=SEQUENCE EncryptionAuthenticationAndIntegrity, CHOICE NonStandardParameter, VideoCapability, AudioCapability, DataApplicationCapability, </pre>
<pre> H222LogicalChannelParameters { resourceID subChannelID pcr-pid programDescriptors streamDescriptors ... } </pre>	<pre> ::=SEQUENCE INTEGER (0..65535), INTEGER (0..8191), INTEGER (0..8191) OPTIONAL, OCTET STRING OPTIONAL, OCTET STRING OPTIONAL, </pre>
<pre> H223LogicalChannelParameters { adaptationLayerType { nonStandard al1Framed al1NotFramed al2WithoutSequenceNumbers al2WithSequenceNumbers al3 { controlFieldOctets sendBufferSize }, ..., al1M al2M al3M }, segmentableFlag ... } </pre>	<pre> ::=SEQUENCE CHOICE NonStandardParameter, NULL, NULL, NULL, NULL, SEQUENCE INTEGER (0..2), INTEGER (0..16777215) -- units octets H223AL1MParameters, H223AL2MParameters, H223AL3MParameters BOOLEAN, </pre>
<pre> H223AL1MParameters { transferMode { framed unframed } } </pre>	<pre> ::=SEQUENCE CHOICE NULL, NULL, </pre>

```

    ...
},
headerFEC                                CHOICE
{
    sebch16-7                             NULL,
    golay24-12                             NULL,
    ...
},
crcLength                                CHOICE
{
    crc4bit                               NULL,
    crc12bit                              NULL,
    crc20bit                              NULL,
    crc28bit                              NULL,
    ...
},
repcCodeRate                             INTEGER (8..32),

arqType                                  CHOICE
{
    noArq                                  NULL,
    typeIArq                              H223AnnexCArqParameters,
    typeIIArq                             H223AnnexCArqParameters,
    ...
},
alpduInterleaving                        BOOLEAN,
alsduSplitting                          BOOLEAN,
...
}

H223AL2MParameters                       ::=SEQUENCE
{
    headerFEC                              CHOICE
    {
        sebch16-5                         NULL,
        golay24-12                         NULL,
        ...
    },
    alpduInterleaving                      BOOLEAN,
    ...
}

H223AL3MParameters                       ::=SEQUENCE
{
    headerFormat                           CHOICE
    {
        sebch16-7                         NULL,
        golay24-12                         NULL,
        ...
    },
    crcLength                              CHOICE
    {
        crc4bit                           NULL,
        crc12bit                           NULL,
        crc20bit                           NULL,
        crc28bit                           NULL,
        ...
    },
    ...
}

```

repcCodeRate	INTEGER (8..32),
arqType	CHOICE
{	
noArq	NULL,
typeIArq	H223AnnexCArqParameters,
typeIIArq	H223AnnexCArqParameters,
...	
},	
alpduInterleaving	BOOLEAN,
...	
}	
H223AnnexCArqParameters	::=SEQUENCE
{	
numberOfRetransmissions	CHOICE
{	
finite	INTEGER (0..16),
infinite	NULL,
...	
},	
sendBufferSize	INTEGER (0..16777215), -- units octets
...	
}	
V76LogicalChannelParameters	::=SEQUENCE
{	
hdlcParameters	V76HDLParameters,
suspendResume	CHOICE
{	
noSuspendResume	NULL,
suspendResumewAddress	NULL,
suspendResumewoAddress	NULL,
...	
},	
uIH	BOOLEAN,
mode	CHOICE
{	
eRM	SEQUENCE
{	
windowSize	INTEGER (1..127) ,
recovery	CHOICE
{	
rej	NULL,
sREJ	NULL,
mSREJ	NULL,
...	
},	
...	
},	
uNERM	NULL,
...	
},	
v75Parameters	V75Parameters,
...	
}	

```

V76HDLParameters ::=SEQUENCE
{
    crcLength          CRCLength,
    n401              INTEGER (1..4095),
    loopbackTestProcedure
    BOOLEAN,
    ...
}

CRCLength ::=CHOICE
{
    crc8bit          NULL,
    crc16bit         NULL,
    crc32bit         NULL,
    ...
}

H2250LogicalChannelParameters ::=SEQUENCE
{
    nonStandard      SEQUENCE OF NonStandardParameter OPTIONAL,
    sessionID        INTEGER(0..255),
    associatedSessionID
    INTEGER(1..255) OPTIONAL,
    mediaChannel     TransportAddress OPTIONAL,
    mediaGuaranteedDelivery
    BOOLEAN OPTIONAL,
    mediaControlChannel
    TransportAddress OPTIONAL, -- reverse RTCP channel
    mediaControlGuaranteedDelivery
    BOOLEAN OPTIONAL,
    silenceSuppression
    BOOLEAN OPTIONAL,
    destination      TerminalLabel OPTIONAL,

    dynamicRTPPayloadType
    INTEGER(96..127) OPTIONAL,
    mediaPacketization
    CHOICE
    {
        h261aVideoPacketization
        NULL,
        ...,
        rtpPayloadType
        RTPPayloadType
    } OPTIONAL,
    ...,
    transportCapability
    TransportCapability OPTIONAL,
    redundancyEncoding
    RedundancyEncoding OPTIONAL,
    source           TerminalLabel OPTIONAL}
    RTPPayloadType ::= SEQUENCE
    {
        payloadDescriptor
        CHOICE
        {
            nonStandardIdentifier
            NonStandardParameter,
            rfc-number
            INTEGER (1..32768, ...),
            oid
            OBJECT IDENTIFIER,
            ...
        },
        payloadType
        INTEGER (0..127) OPTIONAL,
        ...
    }
}

RedundancyEncoding ::=SEQUENCE
{
    redundancyEncodingMethod
    RedundancyEncodingMethod,
    secondaryEncoding
    DataType OPTIONAL, -- depends on method
    ...
}

```

TransportAddress	::=CHOICE
{	
unicastAddress	UnicastAddress,
multicastAddress	MulticastAddress,
...	
}	
UnicastAddress	::=CHOICE
{	
iPAddress	SEQUENCE
{	
network	OCTET STRING (SIZE(4)),
tsapIdentifier	INTEGER(0..65535),
...	
},	
iPXAddress	SEQUENCE
{	
node	OCTET STRING (SIZE(6)),
netnum	OCTET STRING (SIZE(4)),
tsapIdentifier	OCTET STRING (SIZE(2)),
...	
},	
iP6Address	SEQUENCE
{	
network	OCTET STRING (SIZE(16)),
tsapIdentifier	INTEGER(0..65535),
...	
},	
netBios	OCTET STRING (SIZE(16)),
iPSourceRouteAddress	SEQUENCE
{	
routing	CHOICE
{	
strict	NULL,
loose	NULL
},	
network	OCTET STRING (SIZE(4)),
tsapIdentifier	INTEGER(0..65535),
route	SEQUENCE OF OCTET STRING (SIZE(4)),
...	
},	
...,	
nsap	OCTET STRING (SIZE(1..20)),
nonStandardAddress	NonStandardParameter
}	
MulticastAddress	::=CHOICE
{	
iPAddress	SEQUENCE
{	
network	OCTET STRING (SIZE(4)),
tsapIdentifier	INTEGER(0..65535),
...	
},	
iP6Address	SEQUENCE
{	
network	OCTET STRING (SIZE(16)),
tsapIdentifier	INTEGER(0..65535),
...	
},	
...,	

```

    nsap
    nonStandardAddress
}
EncryptionSync ::=SEQUENCE
                -- used to supply new key and synchronization point
{
    nonStandard
    synchFlag
    h235Key
    escrowentry
    ...
}
EscrowData ::=SEQUENCE
{
    escrowID
    escrowValue
    ...
}
OpenLogicalChannelAck ::=SEQUENCE
{
    forwardLogicalChannelNumber LogicalChannelNumber,

    reverseLogicalChannelParameters SEQUENCE
    {
        reverseLogicalChannelNumber LogicalChannelNumber,
        portNumber INTEGER (0..65535) OPTIONAL,
        multiplexParameters CHOICE
        {
            h222LogicalChannelParameters H222LogicalChannelParameters,
            -- H.223 parameters are never present in reverse direction
            ...,
            h2250LogicalChannelParameters H2250LogicalChannelParameters
        }
    } OPTIONAL,
    -- Not present for H.223
    ...,
    replacementFor LogicalChannelNumber OPTIONAL
} OPTIONAL,
-- Not present for unidirectional channel request
...,
separateStack NetworkAccessParameters OPTIONAL,
-- for Open requester to establish the stack
forwardMultiplexAckParameters CHOICE
{
    -- H.222 parameters are never present in the Ack
    -- H.223 parameters are never present in the Ack
    -- V.76 parameters are never present in the Ack
    h2250LogicalChannelAckParameters H2250LogicalChannelAckParameters,
    ...
} OPTIONAL,
encryptionSync EncryptionSync OPTIONAL -- used only by Master
}
OpenLogicalChannelReject ::=SEQUENCE
{
    forwardLogicalChannelNumber LogicalChannelNumber,
    cause CHOICE
    {

```



```

        unspecified                NULL,
        unsuitableReverseParameters  NULL,
        dataTypeNotSupported         NULL,
        dataTypeNotAvailable         NULL,
        unknownDataType              NULL,
        dataTypeALCombinationNotSupported NULL,
        ...,
        multicastChannelNotAllowed    NULL,
        insufficientBandwidth         NULL,
        separateStackEstablishmentFailed NULL,
        invalidSessionID              NULL,
        masterSlaveConflict           NULL,
        waitForCommunicationMode      NULL,
        invalidDependentChannel       NULL,
        replacementForRejected        NULL
    },
    ...
}

OpenLogicalChannelConfirm ::=SEQUENCE
{
    forwardLogicalChannelNumber      LogicalChannelNumber,
    ...
}

H2250LogicalChannelAckParameters ::=SEQUENCE
{
    nonStandard                      SEQUENCE OF NonStandardParameter OPTIONAL,
    sessionID                        INTEGER(1..255) OPTIONAL,
    mediaChannel                      TransportAddress OPTIONAL,
    mediaControlChannel              TransportAddress OPTIONAL, -- forward RTCP channel
    dynamicRTPPayloadType            INTEGER(96..127) OPTIONAL, -- used only by the master
                                     -- or MC
    ...,
    flowControlToZero                BOOLEAN,
    portNumber                        INTEGER (0..65535) OPTIONAL
}

CloseLogicalChannel ::=SEQUENCE
{
    forwardLogicalChannelNumber      LogicalChannelNumber,
    source                            CHOICE
    {
        user                          NULL,
        lcse                           NULL
    },
    ...,
    reason                            CHOICE
    {
        unknown                       NULL,
        reopen                         NULL,
        reservationFailure             NULL,
        ...
    }
}

CloseLogicalChannelAck ::=SEQUENCE
{
    forwardLogicalChannelNumber      LogicalChannelNumber,
    ...
}

```

```

RequestChannelClose ::=SEQUENCE
{
    forwardLogicalChannelNumber LogicalChannelNumber,
    ...,
    qosCapability OPTIONAL,
    reason CHOICE
    {
        unknown NULL,
        normal NULL,
        reopen NULL,
        reservationFailure NULL,
        ...
    }
}

RequestChannelCloseAck ::=SEQUENCE
{
    forwardLogicalChannelNumber LogicalChannelNumber,
    ...
}

RequestChannelCloseReject ::=SEQUENCE
{
    forwardLogicalChannelNumber LogicalChannelNumber,
    cause CHOICE
    {
        unspecified NULL,
        ...
    },
    ...
}

RequestChannelCloseRelease ::=SEQUENCE
{
    forwardLogicalChannelNumber LogicalChannelNumber,
    ...
}

=====
-- H.223 multiplex table definitions
=====

MultiplexEntrySend ::=SEQUENCE
{
    sequenceNumber SequenceNumber,
    multiplexEntryDescriptors SET SIZE (1..15) OF MultiplexEntryDescriptor,
    ...
}

MultiplexEntryDescriptor ::=SEQUENCE
{
    multiplexTableEntryNumber MultiplexTableEntryNumber,
    elementList SEQUENCE SIZE (1..256) OF MultiplexElement OPTIONAL
}

MultiplexElement ::=SEQUENCE
{
    type CHOICE
    {

```

<pre> logicalChannelNumber subElementList }, repeatCount { finite untilClosingFlag } } </pre>	<pre> INTEGER(0..65535), SEQUENCE SIZE (2..255) OF MultiplexElement CHOICE INTEGER (1..65535), -- repeats of type NULL -- used for last element </pre>
<pre> MultiplexTableEntryNumber </pre>	<pre> ::=INTEGER (1..15) </pre>
<pre> MultiplexEntrySendAck { sequenceNumber multiplexTableEntryNumber ... } </pre>	<pre> ::=SEQUENCE SequenceNumber, SET SIZE (1..15) OF MultiplexTableEntryNumber, </pre>
<pre> MultiplexEntrySendReject { sequenceNumber rejectionDescriptions ... } </pre>	<pre> ::=SEQUENCE SequenceNumber, SET SIZE (1..15) OF MultiplexEntryRejectionDescriptions, </pre>
<pre> MultiplexEntryRejectionDescriptions { multiplexTableEntryNumber cause { unspecifiedCause descriptorTooComplex ... }, ... } </pre>	<pre> ::=SEQUENCE MultiplexTableEntryNumber, CHOICE NULL, NULL, </pre>
<pre> MultiplexEntrySendRelease { multiplexTableEntryNumber ... } </pre>	<pre> ::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber, </pre>
<pre> RequestMultiplexEntry { entryNumbers ... } </pre>	<pre> ::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber, </pre>
<pre> RequestMultiplexEntryAck { entryNumbers ... } </pre>	<pre> ::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber, </pre>
<pre> RequestMultiplexEntryReject { entryNumbers rejectionDescriptions </pre>	<pre> ::=SEQUENCE SET SIZE (1..15) OF MultiplexTableEntryNumber, SET SIZE (1..15) OF RequestMultiplexEntryRejectionDescriptions, </pre>

```

}
...
}
RequestMultiplexEntryRejectionDescriptions ::=SEQUENCE
{
    multiplexTableEntryNumber MultiplexTableEntryNumber,
    cause CHOICE
    {
        unspecifiedCause NULL,
        ...
    },
    ...
}

RequestMultiplexEntryRelease ::=SEQUENCE
{
    entryNumbers SET SIZE (1..15) OF MultiplexTableEntryNumber,
    ...
}

-----
-- Request mode definitions
-----

-- RequestMode is a list, in order or preference, of modes that a terminal would like
-- to have transmitted to it

RequestMode ::=SEQUENCE
{
    sequenceNumber SequenceNumber,
    requestedModes SEQUENCE SIZE (1..256) OF ModeDescription,
    ...
}

RequestModeAck ::=SEQUENCE
{
    sequenceNumber SequenceNumber,
    response CHOICE
    {
        willTransmitMostPreferredMode NULL,
        willTransmitLessPreferredMode NULL,
        ...
    },
    ...
}

RequestModeReject ::=SEQUENCE
{
    sequenceNumber SequenceNumber,
    cause CHOICE
    {
        modeUnavailable NULL,
        multipointConstraint NULL,
        requestDenied NULL,
        ...
    },
    ...
}

```

```

RequestModeRelease                               ::=SEQUENCE
{
    ...
}

-----
-- Request mode definitions: Mode description
-----

ModeDescription                                 ::=SET SIZE (1..256) OF ModeElement

ModeElement                                     ::= SEQUENCE
{
    type                                         CHOICE
    {
        nonStandard                             NonStandardParameter,
        videoMode                               VideoMode,
        audioMode                               AudioMode,
        dataMode                                DataMode,
        encryptionMode                          EncryptionMode,
        ...,
        h235Mode                                H235Mode
    },
    h223ModeParameters                          H223ModeParameters OPTIONAL,
    ...,
    v76ModeParameters                          V76ModeParameters OPTIONAL,
    h2250ModeParameters                        H2250ModeParameters OPTIONAL
}

H235Mode                                       ::=SEQUENCE
{
    encryptionAuthenticationAndIntegrity       EncryptionAuthenticationAndIntegrity,
    mediaMode                                  CHOICE
    {
        nonStandard                             NonStandardParameter,
        videoMode                               VideoMode,
        audioMode                               AudioMode,
        dataMode                                DataMode,
        ...
    },
    ...
}

H223ModeParameters                            ::=SEQUENCE
{
    adaptationLayerType                        CHOICE
    {
        nonStandard                             NonStandardParameter,
        al1Framed                               NULL,
        al1NotFramed                           NULL,
        al2WithoutSequenceNumbers              NULL,
        al2WithSequenceNumbers                 NULL,
        al3                                     SEQUENCE
        {
            controlFieldOctets                  INTEGER(0..2),
            sendBufferSize                       INTEGER(0..16777215)    -- units octets
        },
        ...,
}

```

```

        al1M
        al2M
        al3M
    },
    segmentableFlag
    ...
}

V76ModeParameters ::=CHOICE
{
    suspendResumewAddress
    suspendResumewoAddress
    ...
}

H2250ModeParameters ::=SEQUENCE
{
    redundancyEncodingMode
    ...
}

RedundancyEncodingMode ::=SEQUENCE
{
    redundancyEncodingMethod
    secondaryEncoding
    {
        nonStandard
        audioData
        ...
    } OPTIONAL,
    ...
}

=====
-- Request mode definitions: Video modes
=====

VideoMode ::=CHOICE
{
    nonStandard
    h261VideoMode
    h262VideoMode
    h263VideoMode
    is11172VideoMode
    ...
}

H261VideoMode ::=SEQUENCE
{
    resolution
    {
        qcif
        cif
    },
    bitRate
    stillImageTransmission
    ...
}

```

```

H262VideoMode                               ::=SEQUENCE
{
    profileAndLevel                           CHOICE
    {
        profileAndLevel-SPatML                NULL,
        profileAndLevel-MPatLL                NULL,
        profileAndLevel-MPatML                NULL,
        profileAndLevel-MPatH-14              NULL,
        profileAndLevel-MPatHL                NULL,
        profileAndLevel-SNRatLL               NULL,
        profileAndLevel-SNRatML               NULL,
        profileAndLevel-SpatialatH-14        NULL,
        profileAndLevel-HPatML                NULL,
        profileAndLevel-HPatH-14              NULL,
        profileAndLevel-HPatHL                NULL,
        ...
    },
    videoBitRate                              INTEGER(0..1073741823) OPTIONAL, -- units 400 bit/s
    vbvBufferSize                             INTEGER(0..262143) OPTIONAL,   -- units 16 384 bits
    samplesPerLine                             INTEGER(0..16383) OPTIONAL,   -- units samples/line
    linesPerFrame                              INTEGER(0..16383) OPTIONAL,   -- units lines/frame
    framesPerSecond                            INTEGER(0..15) OPTIONAL,    -- frame_rate_code
    luminanceSampleRate                       INTEGER(0..4294967295) OPTIONAL, -- units samples/sec
    ...
}
H263VideoMode                               ::=SEQUENCE
{
    resolution                                CHOICE
    {
        sqcif                                  NULL,
        qcif                                    NULL,
        cif                                      NULL,
        cif4                                    NULL,
        cif16                                   NULL,
        ...
    },
    bitRate                                    INTEGER (1..19200),          -- units 100 bit/s
    unrestrictedVector                        BOOLEAN,
    arithmeticCoding                          BOOLEAN,
    advancedPrediction                        BOOLEAN,
    pbFrames                                  BOOLEAN,
    ...,
    errorCompensation                         BOOLEAN,
    enhancementLayerInfo                      EnhancementLayerInfo OPTIONAL,
    h263Options                               H263Options OPTIONAL
}
IS11172VideoMode                            ::=SEQUENCE
{
    constrainedBitstream                       BOOLEAN,
    videoBitRate                              INTEGER(0..1073741823) OPTIONAL, -- units 400 bit/s
    vbvBufferSize                             INTEGER(0..262143) OPTIONAL,   -- units 16 384 bits
    samplesPerLine                             INTEGER(0..16383) OPTIONAL,   -- units samples/line
    linesPerFrame                              INTEGER(0..16383) OPTIONAL,   -- units lines/frame
    pictureRate                                INTEGER(0..15) OPTIONAL,
    luminanceSampleRate                       INTEGER(0..4294967295) OPTIONAL, -- units samples/sec
    ...
}

```

-- =====
-- Request mode definitions: Audio modes
-- =====

```
AudioMode ::=CHOICE
{
    nonStandard NonStandardParameter,
    g711Alaw64k NULL,
    g711Alaw56k NULL,
    g711Ulaw64k NULL,
    g711Ulaw56k NULL,

    g722-64k NULL,
    g722-56k NULL,
    g722-48k NULL,

    g728 NULL,
    g729 NULL,
    g729AnnexA NULL,

    g7231 CHOICE
    {
        noSilenceSuppressionLowRate NULL,
        noSilenceSuppressionHighRate NULL,
        silenceSuppressionLowRate NULL,
        silenceSuppressionHighRate NULL
    },

    is11172AudioMode IS11172AudioMode,
    is13818AudioMode IS13818AudioMode,

    ...,
    g729wAnnexB INTEGER(1..256),
    g729AnnexAwAnnexB INTEGER(1..256),
    g7231AnnexCMode G7231AnnexCMode,
    gsmFullRate GSMAudioCapability,
    gsmHalfRate GSMAudioCapability,
    gsmEnhancedFullRate GSMAudioCapability
}

IS11172AudioMode ::=SEQUENCE
{
    audioLayer CHOICE
    {
        audioLayer1 NULL,
        audioLayer2 NULL,
        audioLayer3 NULL
    },

    audioSampling CHOICE
    {
        audioSampling32k NULL,
        audioSampling44k1 NULL,
        audioSampling48k NULL
    },
}
```



```

    multichannelType          CHOICE
    {
        singleChannel          NULL,
        twoChannelStereo      NULL,
        twoChannelDual         NULL
    },
    bitRate                    INTEGER (1..448),          -- units kbit/s
    ...
}

IS13818AudioMode            ::=SEQUENCE
{
    audioLayer                CHOICE
    {
        audioLayer1           NULL,
        audioLayer2           NULL,
        audioLayer3           NULL
    },
    audioSampling              CHOICE
    {
        audioSampling16k      NULL,
        audioSampling22k05    NULL,
        audioSampling24k      NULL,
        audioSampling32k      NULL,
        audioSampling44k1     NULL,
        audioSampling48k      NULL
    },
    multichannelType          CHOICE
    {
        singleChannel          NULL,
        twoChannelStereo      NULL,
        twoChannelDual         NULL,
        threeChannels2-1      NULL,
        threeChannels3-0      NULL,
        fourChannels2-0-2-0   NULL,
        fourChannels2-2       NULL,
        fourChannels3-1       NULL,
        fiveChannels3-0-2-0   NULL,
        fiveChannels3-2       NULL
    },
    lowFrequencyEnhancement    BOOLEAN,
    multilingual                BOOLEAN,
    bitRate                     INTEGER (1..1130),        -- units kbit/s
    ...
}

G7231AnnexCMode            ::= SEQUENCE
{
    maxAI-sduAudioFrames      INTEGER (1..256),
    silenceSuppression         BOOLEAN,
    g723AnnexCAudioMode       SEQUENCE
    {

```

```

        highRateMode0    INTEGER (27..78),           -- units octets
        highRateMode1    INTEGER (27..78),           -- units octets
        lowRateMode0     INTEGER (23..66),           -- units octets
        lowRateMode1     INTEGER (23..66),           -- units octets
        sidMode0         INTEGER (6..17),            -- units octets
        sidMode1         INTEGER (6..17),            -- units octets
        ...
    },
    ...
}

```

```

-- =====
-- Request mode definitions: Data modes
-- =====

```

```

DataMode ::=SEQUENCE
{
    application CHOICE
    {
        nonStandard NonStandardParameter,
        t120 DataProtocolCapability,
        dsm-cc DataProtocolCapability,
        userData DataProtocolCapability,
        t84 DataProtocolCapability,
        t434 DataProtocolCapability,
        h224 DataProtocolCapability,
        nlpid SEQUENCE
        {
            nlpidProtocol DataProtocolCapability,
            nlpidData OCTET STRING
        },
        dsvdControl NULL,
        h222DataPartitioning DataProtocolCapability,
        ...,
        t30fax DataProtocolCapability,
        t140 DataProtocolCapability,
        t38fax SEQUENCE
        {
            t38FaxProtocol DataProtocolCapability,
            t38FaxProfile T38FaxProfile
        }
    },
    bitRateINTEGER (0..4294967295), -- units 100 bit/s
    ...
}

```

```

-- =====
-- Request mode definitions: Encryption modes
-- =====

```

```

EncryptionMode ::=CHOICE
{
    nonStandard NonStandardParameter,
    h233Encryption NULL,
    ...
}

```

-- =====
-- Round Trip Delay definitions
-- =====

RoundTripDelayRequest ::=SEQUENCE
{
 sequenceNumber **SequenceNumber,**
 ...
}

RoundTripDelayResponse ::=SEQUENCE
{
 sequenceNumber **SequenceNumber,**
 ...
}

-- =====
-- Maintenance Loop definitions
-- =====

MaintenanceLoopRequest ::=SEQUENCE
{
 type **CHOICE**
 {
 systemLoop **NULL,**
 mediaLoop **LogicalChannelNumber,**
 logicalChannelLoop **LogicalChannelNumber,**
 ...
 },
 ...
}

MaintenanceLoopAck ::=SEQUENCE
{
 type **CHOICE**
 {
 systemLoop **NULL,**
 mediaLoop **LogicalChannelNumber,**
 logicalChannelLoop **LogicalChannelNumber,**
 ...
 },
 ...
}

MaintenanceLoopReject ::=SEQUENCE
{
 type **CHOICE**
 {
 systemLoop **NULL,**
 mediaLoop **LogicalChannelNumber,**
 logicalChannelLoop **LogicalChannelNumber,**
 ...
 },
 cause **CHOICE**
 {
 canNotPerformLoop **NULL,**
 ...
 },
 ...
}

```

MaintenanceLoopOffCommand ::=SEQUENCE
{
    ...
}

-----
-- Communication Mode definitions
-----

CommunicationModeCommand ::=SEQUENCE
{
    communicationModeTable SET SIZE(1..256) OF CommunicationModeTableEntry,
    ...
}

CommunicationModeRequest ::=SEQUENCE
{
    ...
}

CommunicationModeResponse ::=CHOICE
{
    communicationModeTable SET SIZE(1..256) OF CommunicationModeTableEntry,
    ...
}

CommunicationModeTableEntry ::=SEQUENCE
{
    nonStandard SEQUENCE OF NonStandardParameter OPTIONAL,
    sessionID INTEGER(1..255),
    associatedSessionID INTEGER(1..255) OPTIONAL,

    terminalLabel TerminalLabel OPTIONAL, -- if not present,
                                           -- it refers to all participants
                                           -- in the conference

    sessionDescription BMPString (SIZE(1..128)) ,
                       -- Basic ISO/IEC 10646-1 (Unicode)

    dataType CHOICE
    {
        videoData VideoCapability,
        audioData AudioCapability,
        data DataApplicationCapability,
        ...
    },
    mediaChannel TransportAddress OPTIONAL,
    mediaGuaranteedDelivery BOOLEAN OPTIONAL,
    mediaControlChannel TransportAddress OPTIONAL, -- reverse RTCP channel
    mediaControlGuaranteedDelivery BOOLEAN OPTIONAL,
    ...,
    redundancyEncoding RedundancyEncoding OPTIONAL,
    sessionDependency INTEGER (1..255) OPTIONAL,
    destination TerminalLabel OPTIONAL
}

-----
-- Conference Request definitions
-----

ConferenceRequest ::=CHOICE
{

```

```

terminalListRequest      NULL,          -- same as H.230 TCU (term->MC)

makeMeChair             NULL,          -- same as H.230 CCA (term->MC)
cancelMakeMeChair      NULL,          -- same as H.230 CIS (term->MC)

dropTerminal            TerminalLabel, -- same as H.230 CCD(term->MC)

requestTerminalID       TerminalLabel, -- same as TCP (term->MC)

enterH243Password       NULL,          -- same as H.230 TCS1(MC->term)
enterH243TerminalID     NULL,          -- same as H.230 TCS2/TCI
                        -- (MC->term)
enterH243ConferenceID   NULL,          -- same as H.230 TCS3 (MC->term)
...,
enterExtensionAddress   NULL,          -- same as H.230 TCS4 (GW->term)
requestChairTokenOwner  NULL,          -- same as H.230 TCA (term->MC)
requestTerminalCertificate
{
    terminalLabel        TerminalLabel OPTIONAL,
    certSelectionCriteria CertSelectionCriteria OPTIONAL,
    sRandom              INTEGER (1..4294967295) OPTIONAL,
                        -- this is the requester's challenge
    ...
},
broadcastMyLogicalChannel
makeTerminalBroadcaster LogicalChannelNumber,      -- similar to H.230 MCV
sendThisSource          TerminalLabel,             -- similar to H.230 VCB
requestAllTerminalIDs   TerminalLabel,             -- similar to H.230 VCS
remoteMCRequest         NULL,
                        RemoteMCRequest

}

CertSelectionCriteria    ::=SEQUENCE SIZE (1..16) OF Criteria

Criteria
{
    field                OBJECT IDENTIFIER, -- may include certificate type
    value                OCTET STRING (SIZE(1..65535)),
    ...
}

TerminalLabel           ::=SEQUENCE
{
    mcuNumber            McuNumber,
    terminalNumber       TerminalNumber,
    ...
}

McuNumber               ::=INTEGER(0..192)
TerminalNumber          ::=INTEGER(0..192)

-- =====
-- Conference Response definitions
-- =====

ConferenceResponse      ::=CHOICE
{
    mCTerminalIDResponse SEQUENCE                -- response to TCP(same as TIP)
    {
        terminalLabel     TerminalLabel,         -- sent by MC only
    }
}

```

terminalID	TerminalID,	
...		
},		
terminalIDResponse	SEQUENCE	-- response to TCS2 or TCI
{		-- same as IIS
terminalLabel	TerminalLabel,	-- (term->MC)
terminalID	TerminalID,	
...		
},		
conferenceIDResponse	SEQUENCE	-- response to TCS3
{		-- same as IIS
terminalLabel	TerminalLabel,	-- (term->MC)
conferenceID	ConferenceID,	
...		
},		
passwordResponse	SEQUENCE	-- response to TCS1
{		-- same as IIS
terminalLabel	TerminalLabel,	-- (term->MC)
password	Password,	
...		
},		
terminalListResponse	SET SIZE (1..256) OF TerminalLabel,	
videoCommandReject	NULL,	-- same as H.230 VCR
terminalDropReject	NULL,	-- same as H.230 CIR
makeMeChairResponse	CHOICE	-- same as H.230 CCR
{		
grantedChairToken	NULL,	-- same as H.230 CIT
deniedChairToken	NULL,	-- same as H.230 CCR
...		
},		
...,		
extensionAddressResponse	SEQUENCE	-- response to TCS4
{		
extensionAddress	TerminalID,	-- same as IIS (term->GW)
...		
},		
chairTokenOwnerResponse	SEQUENCE	-- response to TCA(same as TIR)
		-- sent by MC only
{		
terminalLabel	TerminalLabel,	
terminalID	TerminalID,	
...		
},		
terminalCertificateResponse	SEQUENCE	
{		
terminalLabel	TerminalLabel OPTIONAL,	
certificateResponse	OCTET STRING (SIZE(1..65535)) OPTIONAL,	
...		
},		
broadcastMyLogicalChannelResponse	CHOICE	
{		
grantedBroadcastMyLogicalChannel	NULL,	
deniedBroadcastMyLogicalChannel	NULL,	
...		
},		

```

makeTerminalBroadcasterResponse      CHOICE
{
    grantedMakeTerminalBroadcaster  NULL,
    deniedMakeTerminalBroadcaster  NULL,
    ...
},
sendThisSourceResponse              CHOICE
{
    grantedSendThisSource           NULL,
    deniedSendThisSource           NULL,
    ...
},
requestAllTerminalIDsResponse      RequestAllTerminalIDsResponse,
remoteMCResponse                  RemoteMCResponse
}

TerminalID                          ::=OCTET STRING (SIZE(1..128))    -- as per H.230
ConferenceID                        ::=OCTET STRING (SIZE(1..32))
Password                            ::=OCTET STRING (SIZE(1..32))

RequestAllTerminalIDsResponse       ::=SEQUENCE
{
    terminalInformation             SEQUENCE OF TerminalInformation,
    ...
}

TerminalInformation                ::=SEQUENCE
{
    terminalLabel                  TerminalLabel,
    terminalID                    TerminalID,
    ...
}

```

-- Remote MC Request definitions

```

RemoteMCRequest                     ::=CHOICE
{
    masterActivate                 NULL,
    slaveActivate                  NULL,
    deActivate                     NULL,
    ...
}

RemoteMCResponse                   ::=CHOICE
{
    accept                          NULL,
    reject                          CHOICE
    {
        unspecified                NULL,
        functionNotSupported       NULL,
        ...
    },
    ...
}

```

```
MultilinkRequest ::= CHOICE
{
  nonStandard
  callInformation      NonStandardMessage,
                      SEQUENCE
  {
    numberOfAdditionalConnections INTEGER (1..65535),
    ...
  },
  addConnection       SEQUENCE
  {
    sequenceNumber     SequenceNumber, -- unique ID of request
    diallingInformation
                      DiallingInformation,
    ...
  },
  removeConnection    SEQUENCE
  {
    connectionIdentifier
                      ConnectionIdentifier,
    ...
  },
  maximumHeaderInterval
  SEQUENCE
  {
    requestType        CHOICE
    {
      currentIntervalInformation
                      NULL,
      requestedInterval
                      INTEGER (0..65535), -- max header interval, milliseconds
      ...
    },
    ...
  },
  ...
}

MultilinkResponse ::= CHOICE
{
  nonStandard
  callInformation      NonStandardMessage,
                      SEQUENCE
  {
    diallingInformation
                      DiallingInformation,
    callAssociationNumber
                      INTEGER (0..4294967295),
    ...
  },
  addConnection       SEQUENCE
  {
    sequenceNumber     SequenceNumber, -- equal to value in request
    responseCode        CHOICE
    {
      accepted          NULL,
      rejected          CHOICE
      {
        connectionsNotAvailable
                      NULL, -- due to any technical reason
        userRejected    NULL,
        ...
      },
    },
  },
}

```



```

        },
        ...
    },
    removeConnection
    {
        connectionIdentifier
        ...
    },
    maximumHeaderInterval
    {
        currentInterval
        ...
    }
    ...
}

MultilinkIndication
{
    nonStandard
    crcDesired
    {
        ...
    },
    excessiveError
    {
        connectionIdentifier
        ...
    },
    ...
}

DiallingInformation
{
    nonStandard
    differential
    infoNotAvailable
    ...
}

DiallingInformationNumber
{
    networkAddress
    subAddress
    networkType
    ...
}

DiallingInformationNetworkType
{
    nonStandard
    n-isdn
    gstn
    ...
}

```

SEQUENCE

ConnectionIdentifier,

SEQUENCE

INTEGER (0..65535), --max header interval, milliseconds

::=CHOICE

NonStandardMessage,
SEQUENCE

SEQUENCE

ConnectionIdentifier,

::=CHOICE

NonStandardMessage,
SET SIZE (1..65535) OF DiallingInformationNumber,
-- list of numbers for all additional channels; only least
-- significant digits different from initial channel's number

INTEGER (1..65535), --maximum No. of additional channels

::=SEQUENCE

NumericString (SIZE (0..40)),
IA5String (SIZE (1..40)) OPTIONAL,
SET SIZE (1..255) OF DiallingInformationNetworkType,

::=CHOICE

NonStandardMessage,
NULL,
NULL,

```

ConnectionIdentifier ::=SEQUENCE
{
    channelTag INTEGER (0..4294967295), -- from H.MULTILINK
    sequenceNumber INTEGER (0..4294967295), -- from H.MULTILINK
    ...
}

```

```

-- =====
-- Command Message definitions
-- =====

```

```

-- =====
-- Command Message: Send Terminal Capability Set
-- =====

```

```

SendTerminalCapabilitySet ::=CHOICE
{
    specificRequest SEQUENCE
    {
        multiplexCapability BOOLEAN,
        capabilityTableEntryNumbers SET SIZE (1..65535) OF CapabilityTableEntryNumber
        OPTIONAL,
        capabilityDescriptorNumbers SET SIZE (1..256) OF CapabilityDescriptorNumber
        OPTIONAL,
        ...
    },
    genericRequest NULL,
    ...
}

```

```

-- =====
-- Command Message: Encryption
-- =====

```

```

EncryptionCommand ::=CHOICE
{
    encryptionSE OCTET STRING, -- per H.233, but no error protection
    encryptionIVRequest NULL, -- requests new IV
    encryptionAlgorithmID SEQUENCE
    {
        h233AlgorithmIdentifier SequenceNumber,
        associatedAlgorithm NonStandardParameter
    },
    ...
}

```

```

-- =====
-- Command Message: Flow Control
-- =====

```

```

FlowControlCommand ::=SEQUENCE
{
    scope CHOICE
    {
        logicalChannelNumber LogicalChannelNumber,
        resourceID INTEGER (0..65535),
        wholeMultiplex NULL
    },
}

```

```

restriction                                CHOICE
{
    maximumBitRate                          INTEGER (0..16777215),    -- units 100 bit/s
    noRestriction                            NULL
},
...
}

```

```

-- =====
-- Command Message: Change or End Session
-- =====

```

```

EndSessionCommand                          ::=CHOICE
{
    nonStandard                              NonStandardParameter,
    disconnect                               NULL,

    gstnOptions                              CHOICE
    {
        telephonyMode                        NULL,
        v8bis                                NULL,
        v34DSVD                              NULL,
        v34DuplexFAX                         NULL,
        v34H324                              NULL,
        ...
    },

    ...,
    isdnOptions                              CHOICE
    {
        telephonyMode                        NULL,
        v140                                 NULL,
        terminalOnHold                       NULL,
        ...
    }
}

```

```

-- =====
-- Command Message: Conference Commands
-- =====

```

```

ConferenceCommand                          ::=CHOICE
{
    broadcastMyLogicalChannel                 LogicalChannelNumber,    -- similar to H.230 MCV
    cancelBroadcastMyLogicalChannel           LogicalChannelNumber,    -- similar to H.230 Cancel-MCV

    makeTerminalBroadcaster                  TerminalLabel,          -- same as H.230 VCB
    cancelMakeTerminalBroadcaster            NULL,                  -- same as H.230 Cancel-VCB

    sendThisSource                           TerminalLabel,          -- same as H.230 VCS
    cancelSendThisSource                     NULL,                  -- same as H.230 cancel VCS

    dropConference                           NULL,                  -- same as H.230 CCK
    ...,
    substituteConferenceIDCommand            SubstituteConferenceIDCommand
}

```

```

SubstituteConferenceIDCommand               ::=SEQUENCE
{

```

```

    conferenceIdentifier          OCTET STRING (SIZE(16)),
    ...
}

-----
-- Command Message: Miscellaneous H.230-like commands
-----

MiscellaneousCommand ::=SEQUENCE
{
    logicalChannelNumber        LogicalChannelNumber,
    type                         CHOICE
    {
        equalizeDelay           NULL,                -- same as H.230 ACE
        zeroDelay               NULL,                -- same as H.230 ACZ
        multipointModeCommand   NULL,
        cancelMultipointModeCommand NULL,
        videoFreezePicture      NULL,
        videoFastUpdatePicture  NULL,

        videoFastUpdateGOB      SEQUENCE
        {
            firstGOB            INTEGER (0..17),
            numberOfGOBs        INTEGER (1..18)
        },

        videoTemporalSpatialTradeOff INTEGER (0..31),    -- commands a trade-off value

        videoSendSyncEveryGOB   NULL,
        videoSendSyncEveryGOBCancelNULL,

        ...,
        videoFastUpdateMB       SEQUENCE
        {
            firstGOB            INTEGER (0..255) OPTIONAL,
            firstMB             INTEGER (1..8192) OPTIONAL,
            numberOfMBs         INTEGER (1..8192),
            ...
        },
        maxH223MUXPDUsizes     INTEGER(1..65535),        -- units octets
        encryptionUpdate       EncryptionSync,
        encryptionUpdateRequest EncryptionUpdateRequest,
        switchReceiveMediaOff  NULL,
        switchReceiveMediaOn   NULL,

        progressiveRefinementStart SEQUENCE
        {
            repeatCount         CHOICE
            {
                doOneProgression    NULL,
                doContinuousProgressions NULL,
                doOneIndependentProgression NULL,
                doContinuousIndependentProgressions NULL,
                ...
            },
            ...
        },
        progressiveRefinementAbortOne    NULL,
        progressiveRefinementAbortContinuous NULL
    },
},
},

```

```

    ...
}

KeyProtectionMethod                               ::=SEQUENCE -- specify how the new key is to be protected
{
    secureChannel                                BOOLEAN,
    sharedSecret                                 BOOLEAN,
    certProtectedKey                             BOOLEAN,
    ...
}

```

```

EncryptionUpdateRequest                           ::=SEQUENCE
{
    keyProtectionMethod                          KeyProtectionMethod OPTIONAL,
    ...
}

```

```

-- =====
-- Command Message: H.223 Multiplex Reconfiguration
-- =====

```

```

H223MultiplexReconfiguration                     ::=CHOICE
{
    h223ModeChange                               CHOICE
    {
        toLevel0                                 NULL,
        toLevel1                                 NULL,
        toLevel2                                 NULL,
        toLevel2withOptionalHeader              NULL,
        ...
    },
    h223AnnexADoubleFlag                         CHOICE
    {
        start                                    NULL,
        stop                                     NULL,
        ...
    },
    ...
}

```

```

-- =====
-- Command Message: New ATM virtual channel command
-- =====

```

```

NewATMVCCCommand                               ::=SEQUENCE
{
    resourceID                                   INTEGER(0..65535),
    bitRate                                      INTEGER(1..65535), -- units 64 kbit/s
    bitRateLockedToPCRClock                     BOOLEAN,
    bitRateLockedToNetworkClock                 BOOLEAN,
    aal                                          CHOICE
    {
        aal1                                     SEQUENCE
        {
            clockRecovery                       CHOICE
            {
                nullClockRecovery                NULL,
                srtsClockRecovery                NULL,
            }
        }
    }
}

```

```

        adaptiveClockRecovery  NULL,
        ...
    },
    errorCorrection             CHOICE
    {
        nullErrorCorrection    NULL,
        longInterleaver        NULL,
        shortInterleaver       NULL,
        errorCorrectionOnly     NULL,
        ...
    },
    structuredDataTransfer     BOOLEAN,
    partiallyFilledCells       BOOLEAN,
    ...
},
aal5                           SEQUENCE
{
    forwardMaximumSDUSize      INTEGER (0..65535), -- units octets
    backwardMaximumSDUSize     INTEGER (0..65535), -- units octets
    ...
},
...
},
multiplex                       CHOICE
{
    noMultiplex                NULL,
    transportStream            NULL,
    programStream              NULL,
    ...
},
reverseParameters             SEQUENCE
{
    bitRate                    INTEGER(1..65535), -- units 64 kbit/s
    bitRateLockedToPCRClock    BOOLEAN,
    bitRateLockedToNetworkClock  BOOLEAN,
    multiplex                   CHOICE
    {
        noMultiplex            NULL,
        transportStream        NULL,
        programStream          NULL,
        ...
    },
    ...
},
...
}

```

```

-- =====
-- Indication Message definitions
-- =====

```

```

-- =====
-- Indication Message: Function not understood
-- =====

```

-- This is used to return a request, response or command that is not understood

```

FunctionNotUnderstood         ::=CHOICE
{
    request                    RequestMessage,
    response                    ResponseMessage,
}

```

```

    command                                CommandMessage
}

-- =====
-- Indication Message: Function not supported
-- =====

-- This is used to return a complete request, response or command that is not recognized

FunctionNotSupported                       ::=SEQUENCE
{
    cause                                  CHOICE
    {
        syntaxError                       NULL,
        semanticError                     NULL,
        unknownFunction                   NULL,
        ...
    },
    returnedFunction                       OCTET STRING OPTIONAL,
    ...
}

-- =====
-- Indication Message: Conference
-- =====

ConferenceIndication                       ::=CHOICE
{
    sbeNumber                              INTEGER (0..9),    -- same as H.230 SBE Number
    terminalNumberAssign                   TerminalLabel,  -- same as H.230 TIA
    terminalJoinedConference                TerminalLabel,  -- same as H.230 TIN
    terminalLeftConference                 TerminalLabel,  -- same as H.230 TID
    seenByAtLeastOneOther                  NULL,           -- same as H.230 MIV
    cancelSeenByAtLeastOneOther            NULL,           -- same as H.230 cancel MIV
    seenByAll                              NULL,           -- like H.230 MIV
    cancelSeenByAll                        NULL,           -- like H.230 MIV
    terminalYouAreSeeing                   TerminalLabel,  -- same as H.230 VIN
    requestForFloor                        NULL,           -- same as H.230 TIF
    ...,
    withdrawChairToken                     NULL,           -- same as H.230 CCR
                                                -- MC-> chair
    floorRequested                         TerminalLabel    -- same as H.230 TIF
                                                -- MC-> chair
}

-- =====
-- Indication Message: Miscellaneous H.230-like indication
-- =====

MiscellaneousIndication                    ::=SEQUENCE
{
    logicalChannelNumber                   LogicalChannelNumber,

```

```

type
{
    logicalChannelActive          NULL,          -- same as H.230 AIA and VIA
    logicalChannelInactive       NULL,          -- same as H.230 AIM and VIS

    multipointConference         NULL,
    cancelMultipointConference   NULL,

    multipointZeroComm          NULL,          -- same as H.230 MIZ
    cancelMultipointZeroComm     NULL,          -- same as H.230 cancel MIZ

    multipointSecondaryStatus    NULL,          -- same as H.230 MIS
    cancelMultipointSecondaryStatus NULL,      -- same as H.230 cancel MIS

    videoIndicateReadyToActivate NULL,          -- same as H.230 VIR

    videoTemporalSpatialTradeOff INTEGER (0..31), -- indicates current trade-off
    ...,
    videoNotDecodedMBs         SEQUENCE
    {
        firstMB                 INTEGER (1..8192),
        numberOfMBs             INTEGER (1..8192),
        temporalReference        INTEGER (0..255),
        ...
    },
    transportCapability         TransportCapability
},
...
}

=====
-- Indication Message: Jitter indication
=====

JitterIndication ::=SEQUENCE
{
    scope
    {
        logicalChannelNumber    LogicalChannelNumber,
        resourceID              INTEGER (0..65535),
        wholeMultiplex          NULL
    },
    estimatedReceivedJitterMantissa INTEGER (0..3),
    estimatedReceivedJitterExponent INTEGER (0..7),
    skippedFrameCount          INTEGER (0..15) OPTIONAL,
    additionalDecoderBuffer     INTEGER (0..262143) OPTIONAL, -- 262143 is 2^18 - 1
    ...
}

=====
-- Indication Message: H.223 logical channel skew
=====

H223SkewIndication ::=SEQUENCE
{
    logicalChannelNumber1      LogicalChannelNumber,
    logicalChannelNumber2      LogicalChannelNumber,
    skew                       INTEGER (0..4095), -- units milliseconds
    ...
}

```



```

=====
-- Indication Message: H.225.0 maximum logical channel skew
=====

```

```

H2250MaximumSkewIndication ::=SEQUENCE
{
    logicalChannelNumber1 LogicalChannelNumber,
    logicalChannelNumber2 LogicalChannelNumber,
    maximumSkew INTEGER (0..4095), -- units milliseconds
    ...
}

```

```

=====
-- Indication Message: MC Location indication
=====

```

```

MCLocationIndication ::=SEQUENCE
{
    signalAddress TransportAddress, -- this is the H.323 Call Signalling
    -- address of the entity which
    -- contains the MC
    ...
}

```

```

=====
-- Indication Message: Vendor identification
=====

```

```

VendorIdentification ::=SEQUENCE
{
    vendor NonStandardIdentifier,
    productNumber OCTET STRING (SIZE(1..256)) OPTIONAL, -- per vendor
    versionNumber OCTET STRING (SIZE(1..256)) OPTIONAL, -- per
    -- productNumber
    ...
}

```

```

=====
-- Indication Message: New ATM virtual channel indication
=====

```

```

NewATMVCIndication ::=SEQUENCE
{
    resourceID INTEGER(0..65535),
    bitRate INTEGER(1..65535), -- units 64 kbit/s
    bitRateLockedToPCRClock BOOLEAN,
    bitRateLockedToNetworkClock BOOLEAN,
    aal CHOICE
    {
        aal1 SEQUENCE
        {
            clockRecovery CHOICE
            {
                nullClockRecovery NULL,
                srtsClockRecovery NULL,
                adaptiveClockRecovery NULL,
                ...
            },

```

```

        errorCorrection          CHOICE
        {
            nullErrorCorrection  NULL,
            longInterleaver      NULL,
            shortInterleaver     NULL,
            errorCorrectionOnly   NULL,
            ...
        },
        structuredDataTransfer   BOOLEAN,
        partiallyFilledCells     BOOLEAN,
        ...
    },
    aal5                          SEQUENCE
    {
        forwardMaximumSDUSize    INTEGER (0..65535), -- units octets
        backwardMaximumSDUSize   INTEGER (0..65535), -- units octets
        ...
    },
    ...
},
multiplex                        CHOICE
{
    noMultiplex                 NULL,
    transportStream             NULL,
    programStream               NULL,
    ...
},
...,
reverseParameters              SEQUENCE
{
    bitRate                     INTEGER(1..65535),      -- units 64 kbit/s
    bitRateLockedToPCRClock     BOOLEAN,
    bitRateLockedToNetworkClock BOOLEAN,
    multiplex                   CHOICE
    {
        noMultiplex             NULL,
        transportStream          NULL,
        programStream            NULL,
        ...
    },
    ...
}
}

```

```

-- =====
-- Indication Message: User input
-- =====

```

```

UserInputIndication            ::=CHOICE
{
    nonStandard                 NonStandardParameter,
    alphanumeric                 GeneralString,
    ...,
    userInputSupportIndication CHOICE
    {
        nonStandard             NonStandardParameter,
        basicString              NULL,
        iA5String                NULL,
        generalString            NULL,
        ...
    },
}

```

```

signal                               SEQUENCE
{
  signalType                          IA5String (SIZE (1) ^ FROM ("0123456789#*ABCD!")),
  duration                             INTEGER (1..65535) OPTIONAL, -- milliseconds
  rtp                                  SEQUENCE

  {
    timestamp                          INTEGER (0..4294967295) OPTIONAL,
    expirationTime                    INTEGER (0..4294967295) OPTIONAL,
    logicalChannelNumber              LogicalChannelNumber,
    ...
  } OPTIONAL,
  ...
},
signalUpdate                          SEQUENCE
{
  duration                            INTEGER (1..65535), -- milliseconds
  rtp                                  SEQUENCE
  {
    logicalChannelNumber              LogicalChannelNumber,
    ...
  } OPTIONAL,
  ...
}
}

```

END

ANNEX B

Messages: Semantic definitions

This annex provides semantic definitions and constraints on the syntax elements defined in the previous clause.

MultimediaSystemControlMessage: Is a choice of message types. Messages defined in this Recommendation are classified as request, response, command and indication messages.

RequestMessage: A request message results in an action by the remote terminal and requires an immediate response from it. The nonStandard message may be used to send non-standard requests.

ResponseMessage: A response message is the response to a request message. The nonStandard message may be used to send non-standard responses.

CommandMessage: A command message requires action but no explicit response. The nonStandard message may be used to send non-standard commands.

IndicationMessage: An indication contains information that does not require action or response. The nonStandard message may be used to send non-standard indications.

NonStandardParameter: This may be used to indicate a non-standard parameter. It consists of an identity and the actual parameters, which are coded as an octet string.

NonStandardIdentifier: Is used to identify the type of non-standard parameter. It is either an object identifier, or an H.221 type of identifier that is an octet string consisting of exactly four octets which are country code (octet 1 as in Recommendation T.35 [28]; octet 2*), manufacturer code (next two octets*), *=assigned nationally. The manufacturer codes are the same as those assigned for use in

Recommendation H.320 [22]. H.245 non-standard identifiers may be either "object" type or "h221NonStandard" type at the discretion of the manufacturer defining the non-standard message, as OBJECT IDENTIFIERS and h221NonStandard messages come from non-overlapping spaces and cannot be confused. However, since h221NonStandard messages are also used by Recommendation H.320, such messages come from the same space as H.320 messages, and shall have the same meaning.

B.1 Master Slave Determination messages

This set of messages is used by a protocol to determine which terminal is the master terminal and which is the slave terminal.

B.1.1 Master Slave Determination

This is sent from a MSDSE to a peer MSDSE.

terminalType is a number that identifies different types of terminal, such as, terminals, MCUs and gateways. The allocation of values to terminal types is outside the scope of this Recommendation.

statusDeterminationNumber is a random number in the range $0..2^{24}-1$.

B.1.2 Master Slave Determination Acknowledge

This is used to confirm whether the terminal is the master terminal or the slave terminal, as indicated by decision. When decision is of type master, the terminal receiving this message is the master terminal and when decision is of type slave, it is the slave terminal.

B.1.3 Master Slave Determination Reject

This is used to reject the MasterSlaveDetermination message. When the cause is of type identicalNumbers, the rejection was due to the random numbers being equivalent and the terminal types being the same.

B.1.4 Master Slave Determination Release

This is sent in the case of a timeout.

B.2 Terminal capability messages

This set of messages is for the secure exchange of capabilities between the two terminals.

B.2.1 Overview

The transmitting terminal assigns each individual mode the terminal is capable of operating in a number in a capabilityTable. For example, G.723.1 audio, G.728 audio, and CIF H.263 video would each be assigned separate numbers.

These capability numbers are grouped into AlternativeCapabilitySet structures. Each AlternativeCapabilitySet indicates that the terminal is capable of operating in exactly one mode listed in the set. For example, an AlternativeCapabilitySet listing {G.711, G.723.1, G.728} means that the terminal can operate in any one of those audio modes, but not more than one.

These AlternativeCapabilitySet structures are grouped into simultaneousCapabilities structures. Each simultaneousCapabilities structure indicates a set of modes the terminal is capable of using simultaneously. For example, a simultaneousCapabilities structure containing the two AlternativeCapabilitySet structures {H.261, H.263} and {G.711, G.723.1, G.728} means that the terminal can operate either of the video codecs simultaneously with any one of the audio codecs. The

simultaneousCapabilities set {{H.261}, {H.261, H.263}, {G.711, G.723.1, G.728}} means the terminal can operate two video channels and one audio channel simultaneously: one video channel per Recommendation H.261, another video channel per either Recommendation H.261 or H.263, and one audio channel per either Recommendation G.711, G.723.1, or G.728.

NOTE – The actual capabilities stored in the capabilityTable are often more complex than presented here. For example, each H.263 capability indicates details including ability to support various picture formats at given minimum picture intervals, and ability to use optional coding modes.

The terminal's total capabilities are described by a set of CapabilityDescriptor structures, each of which is a single simultaneousCapabilities structure and a capabilityDescriptorNumber. By sending more than one CapabilityDescriptor, the terminal may signal dependencies between operating modes by describing different sets of modes which it can simultaneously use. For example, a terminal issuing two CapabilityDescriptor structures, one {{H.261, H.263}, {G.711, G.723.1, G.728}} as in the previous example, and the other {{H.262}, {G.711}}, means the terminal can also operate the H.262 video codec, but only with the low-complexity G.711 audio codec.

Terminals may dynamically add capabilities during a communication session by issuing additional CapabilityDescriptor structures, or remove capabilities by sending revised CapabilityDescriptor structures. All terminals shall transmit at least one CapabilityDescriptor structure.

B.2.2 Terminal Capability Set

This message contains information about the terminal's capability to transmit and receive. It also indicates the version of this Recommendation that is in use. It is sent from an outgoing CESE to a peer incoming CESE.

sequenceNumber is used to label instances of TerminalCapabilitySet so that the corresponding response can be identified.

protocolIdentifier is used to indicate the version of this Recommendation that is in use. Annex A lists the object identifiers defined for use by this Recommendation.

multiplexCapability indicates capabilities relating to multiplexing and network adaptation. A terminal shall include multiplexCapability in the first TerminalCapabilitySet sent.

V75Capability indicates the capabilities of the V.75 control entity. The audioHeader indicates the capability of the V.75 audio header.

B.2.2.1 Capability Table

A capability table is a numbered list of capabilities. A terminal shall be capable of everything that it lists in its capability table, but shall not necessarily be capable of simultaneously performing more than one of them.

A TerminalCapabilitySet may contain zero or more CapabilityTableEntry. At the start, no table entries are defined. When a CapabilityTableEntry is received, it replaces the previously received CapabilityTableEntry with the same CapabilityTableEntryNumber. A CapabilityTableEntry without a Capability may be used to remove the previously received CapabilityTableEntry with the same CapabilityTableEntryNumber.

B.2.2.2 Capability Descriptors

CapabilityDescriptors are used to indicate a terminal's capability to transmit and receive. Each CapabilityDescriptor provides an independent statement about the terminal's capabilities.

capabilityDescriptorNumber is used to number CapabilityDescriptors. If a terminal has a preference for the mode it would like to transmit or receive, and wishes to express this when transmitting its

capabilities, it may do so by giving CapabilityDescriptors that relate to its preferred mode or modes small values of capabilityDescriptorNumber.

simultaneousCapabilities is a set of AlternativeCapabilitySet. It is used to list the simultaneous capabilities of the terminal.

An AlternativeCapabilitySet is a sequence of CapabilityTableEntryNumbers. Only those CapabilityTableEntryNumbers that have been defined shall be present in an AlternativeCapabilitySet, although it is possible to define CapabilityTableEntryNumbers and refer to them in the same TerminalCapabilitySet. If a terminal has a preference for the mode it would like to transmit or receive, and wishes to express this when transmitting its capabilities, it may do so by listing elements in AlternativeCapabilitySets in order of decreasing preference.

A terminal shall be capable of simultaneously performing any one capability from each AlternativeCapabilitySet listed in simultaneousCapabilities.

At least one capability descriptor shall have the following structure: There shall be at least one AlternativeCapabilitySet containing only capabilities of a single medium type for each medium type that the terminal can support. This is to ensure that the remote terminal can select a mode of transmission that includes at least one instance of each medium type that the receiver can support.

NOTE 1 – A repetition of a capability in an AlternativeCapabilitySet is redundant and conveys no further information, while the repetition of a capability in different AlternativeCapabilitySets in the same CapabilityDescriptor indicates the possibility of an additional, simultaneous, instance of the particular capability.

NOTE 2 – Terminals that cannot vary the allocation of resources can indicate their capability completely by use of a single CapabilityDescriptor.

B.2.2.3 Capability

The choices receiveVideoCapability, receiveAudioCapability, receiveDataApplicationCapability and receiveUserInputCapability indicate the capability to receive according to the respective VideoCapability, AudioCapability, DataApplicationCapability and UserInputCapability.

The choices transmitVideoCapability, transmitAudioCapability, transmitDataApplicationCapability and transmitUserInputCapability indicate the capability to transmit according to the respective VideoCapability, AudioCapability, DataApplicationCapability and UserInputCapability.

The choices receiveAndTransmitVideoCapability, receiveAndTransmitAudioCapability, receiveAndTransmitDataApplicationCapability and receiveAndTransmitUserInputCapability indicate the capability to receive and transmit according to the respective VideoCapability, AudioCapability, DataApplicationCapability and UserInputCapability. These code points may be useful for indicating that the receive and transmit capabilities are not independent.

The boolean h233EncryptionTransmitCapability, when true, indicates that the terminal supports encryption according to Recommendations H.233 [14] and H.234 [15].

h233IVResponseTime is measured in units of milliseconds, and indicates the minimum time the receiver requires the transmitter to wait after the completion of transmission of an IV message before starting to use the new IV. The means of transmitting the IV is not defined in this Recommendation.

ConferenceCapability indicates various conference capabilities.

h235SecurityCapability indicates the capabilities that the terminal supports according to Recommendation H.235 [16]. The mediaCapability field shall refer to Capability Table Entries that do contain a transmit, receive, or receiveAndTransmit AudioCapability, VideoCapability, DataApplicationCapability, or similar capability indicated by a NonStandardParameter only.

EncryptionAuthenticationAndIntegrity indicates which encryption, authentication, and integrity capabilities are supported for the signalled mediaCapability. mediaCapability defines the supported audio, video, or data algorithms as well as the supported distribution methods (e.g. receive, transmit, or receive and transmit).

The maxPendingReplacementFor parameter indicates the maximum number of open logical channel operations which are allowed to be in the REPLACEMENT PENDING state simultaneously. The REPLACEMENT PENDING state occurs when a logical channel has been established using the replacementFor parameter, but the replaced logical channel has not yet been closed.

B.2.2.4 Multiplex Capabilities

MultiplexCapability indicates capabilities relating to multiplexing and network adaptation. A terminal shall send MultiplexCapability in the first TerminalCapabilitySet sent. Unless stated otherwise, these are capabilities to receive.

H222Capability: indicates multiplexing and network adaptation capabilities that are specific to the multiplex defined in Recommendation H.222.1 [9].

numberOfVCs indicates how many simultaneous ATM Virtual Channels (VCs) can be supported by the terminal. This includes any VCs that transport H.245, T.120, DSM-CC or any other data, and all VCs that carry audiovisual information. It does not include the VC used for Q.2931 signalling [26].

vcCapability is a set, of size equal to the value of numberOfVCs, that indicates the capabilities present for each available VC.

The sequence aal1, when present, indicates the capability for ATM adaptation layer 1, and which of its options, as specified in Recommendation I.363 [25], are supported. The codepoints are defined in Table B.1.

Table B.1/H.245 – ATM Adaptation Layer 1 codepoints

ASN.1 codepoint	Semantic meaning of codepoint
nullClockRecovery	Null source clock frequency recovery method: synchronous circuit transport
srtsClockRecovery	Synchronous residual timestamp source clock frequency recovery method
adaptiveClockRecovery	Adaptive clock source clock frequency recovery method
nullErrorCorrection	No error correction is supported
longInterleaver	The forward error correction method for loss sensitive signal transport is supported
shortInterleaver	The forward error correction method for delay sensitive signal transport is supported
errorCorrectionOnly	The forward error correction method without cell interleaving is supported
structuredDataTransfer	Structured data transfer is supported
partiallyFilledCells	Partially filled cells are supported

The sequence aal5, when present, indicates the capability for ATM adaptation layer 5, and which of its options, as specified in Recommendation I.363 [25], are supported. forwardMaximumSDUSize and backwardMaximumSDUSize indicate the maximum CPCS-SDU size in the forward and reverse directions, measured in octets. Either aal1 or aal5 or both shall be present.

The booleans transportStream and programStream, when equal to true, indicate the capability to support the Transport Stream and Program Stream multiplexes respectively [8].

availableBitRates indicates the bit rate capabilities for the VC. It is a sequence of different bit rates that can be supported, measured in units of 64 kbit/s. Bit rates are listed in decreasing order, that is, the highest bit rate supported is listed first. Supported bit rates can be listed as individual values using the field singleBitRate, or as a rangeOfBitRates between lowerBitRate and higherBitRate, indicating that all values between this lower limit and higher limit, including these limits, are supported. The bit rates indicated are measured at the AAL-SAP.

The sequence allViaGateway, when present, indicates the capability of ATM adaptation layer 1 supported by AAL1/5 conversion gateways. The codepoints are the same as those of sequence aal1. The sequence Q2931Address indicates one or more sets of Q.2931 party number and party subaddress.

H223Capability: indicates capabilities specific to the H.223 multiplex [10].

The boolean transportWithI-frames, when true, indicates that the terminal is capable of sending and receiving control channel messages using LAPM I-frames as defined in V.42 [37].

The booleans videoWithAL1, videoWithAL2, videoWithAL3, audioWithAL1, audioWithAL2, audioWithAL3, dataWithAL1, dataWithAL2 and dataWithAL3, when true, indicate the capability to receive the stated medium type (video, audio, or data) using the stated adaptation layer (AL1, AL2, or AL3).

The integers maximumAL2SDUSize and maximumAL3SDUSize indicate the maximum number of octets in each SDU that the terminal can receive when using adaptation layer types 2 and 3 respectively.

maximumDelayJitter indicates the maximum peak-to-peak multiplexing jitter that the transmitter shall cause. It is measured in milliseconds. Multiplexing jitter is defined as the difference in time of delivery of the first octet of an audio frame when delivered in the multiplexed stream and when it would be delivered at constant bit rate without a multiplex.

h223MultiplexTableCapability: indicates the terminal's ability to receive and process multiplex table entries.

basic indicates that the multiplex can only receive basic MultiplexEntryDescriptors as defined in Recommendation H.223 [10].

enhanced indicates that the multiplex can receive enhanced MultiplexEntryDescriptors with the additional parameters defined below.

maximumNestingDepth depth indicates the maximum nesting depth of recursively invoked subElementList fields. MultiplexEntryDescriptors which do not use the subElementList field shall be considered to have a nesting depth of zero.

maximumElementListSize indicates the maximum number of fields in the ASN.1 SEQUENCE.

maximumSubElementListSize indicates the maximum number of subelements in the subElementList.

The boolean maxMUXPDUSizeCapability, when true, indicates that the transmitter is able to restrict the size of the H.223 MUX-PDUs that it transmits. It has no meaning when part of a receive capability.

The boolean nsrpSupport, when true, indicates support of the Annex A/H.324 NSRP mode.

MobileOperationTransmitCapability: indicates the capability transmit the multiplex layers described in Annex A/H.223 and Annex B/H.223.

The boolean h223AnnexA, if true, indicates the terminal can transmit the MUX-PDUs as defined in Annex A/H.223.

The boolean `h223AnnexADoubleFlag`, if true, indicates the terminal can transmit the MUX-PDUs as defined in Annex A/H.223 with its optional double-flag mode.

The boolean `h223AnnexB`, if true, indicates the terminal can transmit the MUX-PDUs as defined in Annex B/H.223.

The boolean `h223AnnexBwithOptionalHeaderField`, if true, indicates the terminal can transmit the MUX-PDU as defined in Annex B/H.223 with its optional header field.

h223AnnexCCapability: indicates the capability to receive and process AL-PDUs as described in Annex C/H.223, with the following condition.

The booleans `videoWithAL1M`, `videoWithAL2M`, `videoWithAL3M`, `audioWithAL1M`, `audioWithAL2M`, `audioWithAL3M`, `dataWithAL1M`, `dataWithAL2M` and `dataWithAL3M`, when true, indicate the capability to receive the stated medium type (video, audio, or data) using the stated adaptation layer (AL1M, AL2M, or AL3M).

`alpduInterleaving`, if true, indicates the capability to receive and process AL-PDU's for which interleaving is applied.

The integer `maximumAL1MPDUSize` indicates the maximum number of octets in each PDU that the terminal can receive when using adaptation layer AL1M.

The integers `maximumAL2MSDUSize` and `maximumAL3MSDUSize` indicate the maximum number of octets in each SDU that the terminal can receive when using adaptation layer AL2M and AL3M, respectively.

V76Capability: indicates capabilities specific to the V.76 multiplex.

The `suspendResumeCapabilitywAddress` indicates the capability of supporting V.76 suspend/resume with an address field. The `suspendResumeCapabilitywoAddress` indicates the capability of supporting V.76 suspend/resume without an address field.

`rejCapability` indicates the capability of the V.76 multiplex error control function to perform reject.

`sREJCapability` indicates the capability of the multiplex error control function to perform selective reject.

`mREJCapability` indicates the capability of the multiplex error control function to perform multiple selective reject.

`crc8bitCapability` is the capability of the multiplex to use 8-bit CRC.

`crc16bitCapability` is the capability of multiplex to use 16-bit CRC.

`crc32bitCapability` is the capability of the multiplex to use 32-bit CRC.

`uihCapability` indicates support of V.76 UIH frames.

`numOfDLCs` indicates the number of DLCs which the V.76 multiplex can support.

`twoOctetAddressFieldCapability` indicates the ability of the V.76 multiplex to support an address field of two octets.

`loopBackTestCapability` indicates the support of loop back per Recommendation V.76.

`n401Capability` indicates the maximum value of N401 described in Recommendation V.76.

`maxWindowSizeCapability` indicates the maximum window size the V.76 multiplex can support.

H2250Capability: indicates capabilities specific to the H.225.0 media packetization layer.

`maximumAudioDelayJitter` indicates the maximum peak-to-peak delivery of audio packets to the transport layer that the transmitter shall cause. It is measured in milliseconds.

receiveMultipointCapability indicates the receive capabilities of a terminal in a multipoint conference.

transmitMultipointCapability indicates the transmit capabilities of a terminal in a multipoint conference.

receiveAndTransmitMultipointCapability indicates the receive and transmit capabilities of a terminal in a multipoint conference.

mcCapability indicates the ability of a terminal to act as an MC in a centralized or distributed conference.

rtcpVideoControlCapability indicates a terminal's ability to process both RTCP Full Intra Request (FIR) and Negative Acknowledgement (NACK) messages.

MediaPacketizationCapability indicates which optional media packetization scheme are supported by the endpoint.

h261aVideoPacketization indicates that the H261 alternative RTP payload format described in Recommendation H.225.0 is in use.

rtpPayloadType indicates the RTP payload packetization schemes supported by the endpoint as follows.

payloadDescriptor identifies the semantics associated with the payloadType: if the rfc-number is chosen, it indicates the official document of the IETF in which the payload format is defined; obsoleted RFCs should not be referenced here. If the oid component is chosen, this identifies a payload format specified as part of a Recommendation defined by the ITU or an International Standard defined by the ISO and registered in the respective document under this Object Identifier. This applies equally well to both capability exchange and opening logical channels.

payloadType may be included to indicate which payload type is associated with this format. If used in capability exchange, the payloadType shall be set to a statically assigned payload type if one exists for this payload format. Otherwise, the payloadType shall be omitted. If used in conjunction with OpenLogicalChannel, the payloadType shall indicate the payload type value to be used. If a dynamic payload type is used, the values of the payloadType field and the dynamicPayloadType field shall match.

TransportCapability indicates optional transport capabilities such as quality of service and median channel type capabilities.

redundancyEncodingCapability indicates which redundancy encoding modes are supported (if any). For each capability entry, the redundancyEncodingMethod specifies the type of encoding to be used, the primary encoding, and which secondary encodings are supported for this primary encoding. The choice of encoding schemes depends on the mode selected. rtpAudioRedundancyEncoding refers to the audio redundancy encoding; if this mode is the selected redundancyEncodingMethod, only CapabilityEntryNumbers referring to audio encodings are valid. rtpH263VideoRedundancyEncoding indicates that video redundancy coding according to H.263+ Annex N is possible or that a logical channel shall be opened using video redundancy coding. Additional parameters are provided as follows:

numberOfThreads indicates the maximum number of the threads the sender/receiver is able to support when used during capability exchange; it contains the actual number of threads for a specific stream when opening a logical channel.

framesBetweenSyncPoints defines the maximum number of video frames that may be transmitted (summed across all threads) between two synchronization points of all threads during capability exchange; defines the actual number of frames for a specific stream for OpenLogicalChannel.

frameToThreadMapping defines which modes are supported by a sender/receiver during capability exchange and which mode is to be used when opening a logical channel: "round-robin" indicates that frames are assigned in a round-robin fashion to the threads, with the first frame after a synchronization point being assigned to thread 0, the second to thread 1, and so forth. The "custom" format allows to specify arbitrary mappings of frames to threads; during caps exchange, support for custom format is indicated by choosing this component and encoding an arbitrary (possibly empty) SEQUENCE. Support for custom formats implies support for round-robin mappings.

containedThreads applies only to commands that open logical channels: this parameter then indicates which of the threads are transmitted in logical channel to be opened. A logical channel may contain any number of 1 through 15 threads, however, two logical channels shall not specify to contain the same thread.

In case of rtpH263VideoRedundancyEncoding, the secondaryEncoding parameter shall not be present; this also applies to the H2250ModeParameters and the RedundancyEncoding ASN.1 structures of Recommendation H.245.

When a logical channel for video redundancy coding is opened, the logical channel containing thread 0 shall be opened first, and this logical channel shall be referenced by all other logical channels by means of the forwardLogicalChannelDependency parameter in the OpenLogicalChannel command.

LogicalChannelSwitchingCapability indicates the ability of a receiver to switch which stream (e.g. which logical channel) is being rendered based on the switchReceiveMedia on and off commands.

t120DynamicPortCapability indicates that the endpoint can place a T.120 [32] call to a dynamic transport address instead of the standard well-known port address as defined in Recommendation T.123 [33].

multipointCapability: indicates a terminal's capabilities specific to multipoint.

multicastCapability indicates the ability of a terminal to multicast audio or video traffic.

multiUniCastConference indicates the ability of a terminal to participate in a multiUniCast conference.

mediaDistributionCapability: indicates a terminal's capabilities for transmission and reception of media in a multipoint conference. Centralized Control and Audio shall be TRUE for H.323 terminals. If Video is supported, the Centralized Video shall be set TRUE. If T.120 is supported, the Centralized Data T.120 Data Application Capability shall be present.

Centralized and distributed control, audio, and video, indicate the ability of a terminal to participate in a conference with those media distribution types. Centralized and distributed data indicate the ability of a terminal to participate in conference with those media distribution types for the specific Data Application Protocol. MediaDistributionCapability is a sequence to allow for the definition of simultaneous capabilities (e.g. centralized audio with distributed video or centralized video with distributed audio, or specific data capabilities per a Data Application Protocol).

qOSCapabilities indicate quality of service capabilities such as RSVPPParameters and ATMPParameters parameters.

mediaChannelCapabilities indicate what transports the media channel may be carried on. IP-UDP indicates that the endpoint supports transporting the media channel over an the IP network layer and a UDP transport layer. IP-TCP indicates that the endpoint supports transporting the media channel over an IP network layer and a TCP transport layer. atm-AAL5-UNIDIR indicates that the endpoint support transporting the media channel over an ATM AAL5 unidirectional virtual circuit.

atm-AAL5-BIDIR indicates that the endpoint support transporting the media channel over an ATM AAL5 bidirectional virtual circuit.

RSVPParameters indicate specific parameter information about the RSVP protocol.

ATMPParameters indicate specific parameter information about an ATM virtual circuit.

QosMode indicates whether the mode is a guaranteed quality of service or controlled load mode where no upper bound on end-to-end delay is enforced.

B.2.2.5 Video Capabilities

This indicates video capabilities. The indication of more than a single capability within a single VideoCapability does not indicate simultaneous processing capability. Simultaneous processing capability can be indicated by instances of VideoCapability in different AlternativeCapabilitySets in a single CapabilityDescriptor.

H261VideoCapability: indicates H.261 [18] capabilities.

If present, qcifMPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of QCIF pictures, and if not present, no capability for QCIF pictures is indicated.

If present, cifMPI indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of CIF pictures, and if not present, no capability for CIF pictures is indicated.

The boolean temporalSpatialTradeOffCapability, when true, indicates that the encoder is able to vary its trade-off between temporal and spatial resolution as commanded by the remote terminal. It has no meaning when part of a receive capability.

maxBitRate indicates the maximum bit rate in units of 100 bit/s at which a transmitter can transmit video or a receiver can receive video.

stillImageTransmission indicates the capability for still images as specified in Annex D/H.261.

H262VideoCapability: indicates H.262 [19] capabilities.

The list of booleans indicates the capability of processing the particular profiles and levels: a value of true indicates that such operation is possible, while a value of false indicates that such operation is not possible. An encoder shall produce bitstreams compliant to the specifications of a profile and level for which it has indicated capability, but also within the limitations imposed by the optional fields (see below). A decoder shall be able to accept all bit streams conforming to a profile and level for which it has indicated capability, provided it is within the limitations indicated by the optional fields. The optional fields are integers with units defined in Table B.2.

Table B.2/H.245 – Units for H.262 codepoints

ASN.1 codepoint	Units for referenced parameter
videoBitRate	400 bit/s
vbvBufferSize	16 384 bits
samplesPerLine	Samples per line
linesPerFrame	Lines per frame
framesPerSecond	The index, frame_rate_code, into Table 6-4/H.262.
luminanceSampleRate	Samples per second

H263VideoCapability: indicates H.263 [20] capabilities.

If present, `sqcifMPI` indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of SQCIF pictures, and if not present, no capability for SQCIF pictures is indicated.

If present, `qcifMPI` indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of QCIF pictures, and if not present, no capability for QCIF pictures is indicated.

If present, `cifMPI` indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of CIF pictures, and if not present, no capability for CIF pictures is indicated.

If present, `cif4MPI` indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of 4CIF pictures, and if not present, no capability for 4CIF pictures is indicated.

If present, `cif16MPI` indicates the minimum picture interval in units of 1/29.97 for the encoding and/or decoding of 16CIF pictures, and if not present, no capability for 16CIF pictures is indicated.

`maxBitRate` indicates the maximum bit rate in units of 100 bit/s at which a transmitter can transmit video or a receiver can receive video.

The booleans `unrestrictedVector` (see Annex D/H.263), `arithmeticCoding` (see Annex E/H.263), `advancedPrediction` (see Annex F/H.263), and `pbFrames` (see Annex G/H.263), when true, indicate the capability to transmit and/or receive these optional modes defined in the annexes of Recommendation H.263.

The boolean `temporalSpatialTradeOffCapability`, when true, indicates that the encoder is able to vary its trade-off between temporal and spatial resolution as commanded by the remote terminal. It has no meaning when part of a receive capability.

The integer `hrd-B`, when present, indicates the HRD parameter B in Annex B/H.263, and is measured in units of 128 bits. When not present, the default value defined in Annex B/H.263 applies. It is a receiver capability and has no meaning in transmission capability sets.

The integer `bppMaxKb`, when present, indicates the maximum number of bits for one coded picture that the receiver can receive and decode correctly, and is measured in units of 1024 bits. When not present, the default value defined in Recommendation H.263 applies. It is a receiver capability and has no meaning in transmission capability sets.

The following capabilities are intended for use in certain very low frame rate applications such as surveillance applications:

If present, `slowSqcifMPI` indicates the minimum picture interval in units of seconds per frame for the encoding and/or decoding of SQCIF pictures. If not present and `sqcifMPI` is not present, no capability for SQCIF pictures is indicated. If `sqcifMPI` is present, `slowSqcifMPI` shall not be present.

If present, `slowQcifMPI` indicates the minimum picture interval in units of seconds per frame for the encoding and/or decoding of QCIF pictures. If not present and `qcifMPI` is not present, no capability for QCIF pictures is indicated. If `qcifMPI` is present, `slowQcifMPI` shall not be present.

If present, `slowCifMPI` indicates the minimum picture interval in units of seconds per frame for the encoding and/or decoding of CIF pictures. If not present and `cifMPI` is not present, no capability for CIF pictures is indicated. If `cifMPI` is present, `slowCifMPI` shall not be present.

If present, `slowCif4MPI` indicates the minimum picture interval in units of seconds per frame for the encoding and/or decoding of 4CIF pictures. If not present and `cif4MPI` is not present, no capability for 4CIF pictures is indicated. If `cif4MPI` is present, `slowCif4MPI` shall not be present.

If present, `slowCif16MPI` indicates the minimum picture interval in units of seconds per frame for the encoding and/or decoding of 16CIF pictures. If not present and `cif16MPI` is not present, no capability for 16CIF pictures is indicated. If `cif16MPI` is present, `slowCif16MPI` shall not be present.

The values of MPI are applicable when all of the optional modes, for which capability is indicated, are being used, as well as when any combination of them is used. A terminal may signal the capability for a smaller MPI when some options are not used by transmitting another VideoCapability including this smaller MPI and indicating the reduced set of options.

The boolean errorCompensation, when true, indicates the capability to transmit and/or receive feedback information for error compensation as illustrated in Appendix I/H.263. When part of a transmit capability, it indicates the ability of the encoder to process videoNotDecodedMBs indications and compensate errors. When part of a receive capability, it indicates the ability of the decoder to identify erroneous MBs, treat them as not coded, and send appropriate videoNotDecodedMBs indications.

If present, enhancementLayerInfo indicates the capability of the encoder to transmit, or the decoder to receive, bitstreams with the optional scalability mode (see Annex O/H.263). enhancementLayerInfo is a sequence which indicates the configuration parameters of the scalability mode.

If present, H263Options indicates the capability for optional modes of H.263.

EnhancementLayerInfo: indicates capability for the scalability mode of H.263.

baseBitRateConstrained indicates whether the base layer is constrained not to exceed the maximum bit rate in the video capability minus the sum of the maximum bit rate in each of the enhancement options.

When present, snrEnhancement indicates the presence of an snr enhancement layer capability. The set size indicates the number of snrEnhancement layers the terminal is capable of supporting within a single logical channel.

When present, spatialEnhancement indicates the presence of a spatial enhancement layer capability. An enhancement layer bit stream contains a picture size which is either twice the width, or twice the height, or both, of the picture size in the layer which it references. For a terminal to be capable of spatial enhancement in one dimension (width or height), a terminal must also indicate the capability to support the associated custom picture format required in the enhancement layer. The set size indicates the number of spatialEnhancement layers the terminal is capable of supporting within a single logical channel.

When present, bPictureEnhancement indicates the presence of a B pictures enhancement layer capability. The set size indicates the number of bPictureEnhancement layers the terminal is capable of supporting within a single logical channel.

EnhancementOptions inside the bPictureEnhancement sequence indicates which additional options an encoder may transmit or a decoder can receive in the B pictures.

numberOfBPictures indicates the maximum number of B pictures the terminal is capable of supporting between successive pairs of anchor reference pictures used in the prediction of the B pictures. For example, if equal to 2, then two B pictures can be sent between each pair of P pictures or other anchor pictures.

EnhancementOptions: indicates scalability enhancement layer capabilities.

The parameters in EnhancementOptions have the same semantic definitions as the parameters of the same name in H263VideoCapability.

H263Options: indicates capability of additional optional modes of H.263.

advancedIntraCodingMode, when true, indicates the capability to transmit or receive the Advanced INTRA Coding Mode Annex I/H.263.

deblockingFilterMode, when true, indicates the capability to transmit or receive the Deblocking Filter Mode Annex J/H.263.

improvedPBFramesMode, when true, indicates the capability to transmit or receive the Improved PB frames Mode Annex M/H.263.

unlimitedMotionVectors, when true, indicates the capability of the encoder or decoder to support unlimited motion vector range when Unrestricted Motion Vector Mode (see Annex D/H.263) is also indicated. unlimitedMotionVectors shall be FALSE if unrestrictedVector is FALSE in the same H263VideoCapability or H263VideoMode.

fullPictureFreeze, when true, indicates the capability of the encoder to send or the decoder to receive Full Picture Freeze commands as described in Annex L/H.263.

partialPictureFreezeAndRelease, when true, indicates the capability of the encoder to send or the decoder to receive Full Picture Freeze and Release commands as described in Annex L/H.263.

resizingPartPicFreezeAndRelease, when true, indicates the capability of the encoder to send or the decoder to receive the Resizing Partial Picture Freeze and Release commands as described in Annex L/H.263.

fullPictureSnapshot, when true, indicates the capability of the encoder to send or the decoder to receive Full Picture snapshots of the video content as described in Annex L/H.263.

partialPictureSnapshot, when true, indicates the capability of the encoder to send or the decoder to receive Partial Picture Snapshots of the video content as described in Annex L/H.263.

videoSegmentTagging, when true, indicates the capability of the encoder to send or the decoder to receive Video Segment tagging for the video content as described in Annex L/H.263.

progressiveRefinement, when true, indicates the capability of the encoder to send or the decoder to receive Progressive Refinement tagging as described in Annex L/H.263. In addition, when true, the encoder shall respond to the progressive refinement miscellaneous commands doOneProgression, doContinuousProgressions, doOneIndependentProgression, doContinuousIndependentProgressions, progressiveRefinementAbortOne, and progressiveRefinementAbortContinuous. In addition, the encoder shall insert the Progressive Refinement Segment Start Tags and the Progressive Refinement Segment End Tags as defined in the Supplemental Enhancement Information Specification of Annex L/H.263.

NOTE – Progressive Refinement tagging can be sent by an encoder and received by a decoder even when not commanded in a miscellaneous command.

dynamicPictureResizingByFour, when true, indicates the capability of an encoder or decoder to support the picture resizing-by-four (with clipping) submode of the implicit Reference Picture Resampling Mode of Annex P/H.263.

dynamicPictureResizingSixteenthPel, when true, indicates the capability of an encoder or decoder to support resizing a reference picture to any width and height using the implicit Reference Picture Resampling mode of Annex P/H.263 (with clipping).

dynamicWarpingHalfPel, when true, indicates the capability of an encoder or decoder to support the arbitrary picture warping operation within the Reference Picture Resampling mode of Annex P/H.263 (with any fill mode) using half-pixel accuracy warping.

dynamicWarpingSixteenthPel, when true, indicates the capability of an encoder or decoder to support the arbitrary picture warping operation within the Reference Picture Resampling mode of Annex P/H.263 (with any fill mode) using either half-pixel or sixteenth pixel accuracy warping.

If `dynamicPictureResizingSixteenthPel` is true, then `dynamicPictureResizingByFour` shall be true. If `dynamicWarpingSixteenthPel` is true, then `dynamicWarpingHalfPel`, `dynamicPictureResizingByFour`, and `dynamicPictureResizingSixteenthPel` shall be true.

`independentSegmentDecoding`, when true, indicates the capability of an encoder or decoder to support the Independent Segment Decoding Mode of Annex R/H.263.

`slicesInOrder-NonRect`, when true, indicates the capability of an encoder or decoder to support the submode of Slice Structured Mode (see Annex K/H.263) where slices are transmitted in order and contain macroblocks in scanning order of the picture.

`slicesInOrder-Rect`, when true, indicates the capability of an encoder or decoder to support the submode of Slice Structured Mode (see Annex K/H.263) where slices are transmitted in order and the slice occupies a rectangular region of the picture.

`slicesNoOrder-NonRect`, when true, indicates the capability of an encoder or decoder to support the submode of Slice Structured Mode (see Annex K/H.263) where slices contain macroblocks in scanning order of the picture and need not be transmitted in order.

`slicesNoOrder-Rect`, when true, indicates the capability of an encoder or decoder to support the submode of Slice Structured Mode (see Annex K/H.263) where slices occupy a rectangular region of the picture and need not be transmitted in order.

`alternateInterVLCMode`, when true, indicates the capability of an encoder or decoder to support Alternate Inter VLC Mode of Annex S/H.263.

`modifiedQuantizationMode`, when true, indicates the capability of an encoder or decoder to support the Modified Quantization Mode of Annex T/H.263.

`reducedResolutionUpdate`, when true, indicates the capability of an encoder or decoder to support the Reduced Resolution Update mode defined in Annex Q/H.263.

TransparencyParameters: indicate parameters specifying a transparent video layer.

`presentationOrder` indicates the layering of transparent video layers. During capability exchange, the value of `presentationOrder` shall take one of the values 0, 1 and 2: if 0, it indicates that the Reference Picture Background (RPB) type of transparency support as defined in Annex L/H.263 is supported; if 1, it indicates that an externally controlled background picture can be used; and if 2, it indicates that the bitstream can specify use of either the Reference Picture Background transparency or an externally-controlled background picture type of transparency. During logical open channel, the INTEGER value specifies the presentation order. A layer with a higher presentation order shall be layered on top of a layer with a lower presentation order. The `presentationOrder` can be viewed as an axis perpendicular to the screen with direction of increasing parameter towards the viewer.

`offset-x` and `offset-y` indicate the pixel offset, in 1/8 pixels, of the signalled transparent layer to base layer, in units relative to the base layer. When used in a capability, these denote the capability to offset the location of the transparent video layer, and shall have values restricted to 1, 2, 4, or 8, in 1/8 pel units: for example, if the value is 4, the capability to offset the transparent layer in 1/2 pixel increments is indicated.

`scale-x` and `scale-y` indicate a scaling factor to be applied in the corresponding x and y coordinates to the signalled transparent layer before video layering, in units relative to the base layer. In a capability message, they indicate the maximum scale factor that can be applied: 1 indicates rescaling is not supported, 2 indicates it can double the size of the layer or keep it unscalled, 3 indicates it can double it, triple it or keep it unscalled, etc.).

The boolean **`separateVideoBackChannel`** indicates, when true, that the terminal can support the Separate Logical Channel mode: no other video capability shall be indicated in the same

H263VideoCapability: no MPI values shall be present, and all other mode flags and contents have no meaning and shall be false or absent. When sent in a mode request message, **separateVideoBackChannel = true** shall be sent as the only video capability in that **H263VideoMode**, and indicates that the receiver wants to receive a channel containing only H.263 back-channel data. If present in the **OpenLogicalChannel** message, it indicates that the logical channel is for video back-channel messages only and no other H.263 video bitstream shall be delivered by that logical channel.

refPictureSelection: indicates the capability of Reference Picture Selection mode (see Annex N/H.263).

If present, **additionalPictureMemory** indicates the presence of the extra amount of memory, in addition to the amount which can be used by the normal decoder which does not support reference picture selection mode. If not present, it indicates that no information regarding the additional amount of memory which the decoder can use is available for an encoder at the other terminal. If it is indicated in **H263VideoMode**, it indicates the presence of the additional amount of picture memory used for decoding.

sqcifAdditionalPictureMemory indicates that the encoder can send or the decoder can receive an H.263 bitstream which requires the decoder to have the additional memory to store the indicated number of pictures of size SQCIF, or of a smaller size in both horizontal and vertical dimension if custom picture format support for such pictures is indicated in **customPictureFormat**.

qcifAdditionalPictureMemory indicates that the encoder can send or the decoder can receive an H.263 bitstream which requires the decoder to have the additional memory to store the indicated number of pictures of size QCIF, or of a smaller size in both horizontal and vertical dimension if custom picture format support for such pictures is indicated in **customPictureFormat**. The number of picture memories indicated in **qcifAdditionalPictureMemory** shall not be larger than the number of pictures indicated in **sqcifAdditionalPictureMemory** (if present).

cifAdditionalPictureMemory indicates that the encoder can send or the decoder can receive an H.263 bitstream which requires the decoder to have the additional memory to store the indicated number of pictures of size CIF, or of a smaller size in both horizontal and vertical dimension if custom picture format support for such pictures is indicated in **customPictureFormat**. The number of picture memories indicated in **cifAdditionalPictureMemory** shall not be larger than the number of pictures indicated in **sqcifAdditionalPictureMemory**, or **qcifAdditionalPictureMemory** (if present).

cif4AdditionalPictureMemory indicates that the encoder can send or the decoder can receive an H.263 bitstream which requires the decoder to have the additional memory to store the indicated number of pictures of size 4CIF, or of a smaller size in both horizontal and vertical dimension if custom picture format support for such pictures is indicated in **customPictureFormat**. The number of picture memories indicated in **cif4AdditionalPictureMemory** shall not be larger than the number of pictures indicated in **sqcifAdditionalPictureMemory**, **qcifAdditionalPictureMemory**, or **cifAdditionalPictureMemory** (if present).

cif16AdditionalPictureMemory indicates that the encoder can send or the decoder can receive an H.263 bitstream which requires the decoder to have the additional memory to store the indicated number of pictures of size 16CIF, or of a smaller size in both horizontal and vertical dimension if custom picture format support for such pictures is indicated in **customPictureFormat**. The number of picture memories indicated in **cif16AdditionalPictureMemory** shall not be larger than the number of pictures indicated in **sqcifAdditionalPictureMemory**, **qcifAdditionalPictureMemory**, **cifAdditionalPictureMemory**, or **cif4AdditionalPictureMemory** (if present).

bigCpfAdditionalPictureMemory indicates that the encoder can send or the decoder can receive an H.263 bitstream which requires the decoder to have the additional memory to store the indicated number of pictures having a custom picture format of a size indicated in **customPictureFormat** which

are larger than 16CIF in the horizontal or vertical dimension. The number of picture memories indicated in `bigCPFAdditionalPictureMemory` shall not be larger than the number of pictures indicated in `sqcifAdditionalPictureMemory`, `qcifAdditionalPictureMemory`, `cifAdditionalPictureMemory`, `cif4AdditionalPictureMemory`, or `cif16AdditionalPictureMemory` (if present).

`videoMux` indicates, during the capability exchange procedure, that the terminal can support the VideoMux mode illustrated in Annex N/H.263. When true, the encoder or decoder can use a video bitstream containing video back-channel message. If it is indicated in `H263VideoMode`, it indicates that receiving video back-channel messages in VideoMux mode is preferable. When used in `H263VideoMode`, `videoMux` and `separateVideoBackChannel` shall not both be true.

`videoBackChannelSend` indicates which type of video back-channel message is supported by the terminal. If it is indicated in `H263VideoMode`, it indicates the preferred type of back-channel message.

`none` indicates that the encoder is not capable of sending or the decoder is not capable of receiving an H.263 bitstream which contains requests for any back-channel messages to be returned.

`ackMessageOnly` indicates that the encoder is capable of sending or the decoder is capable of receiving an H.263 bitstream which contains requests for only acknowledgement back-channel messages to be returned.

`nackMessageOnly` indicates that the encoder is capable of sending or the decoder is capable of receiving an H.263 bitstream which contains requests for only non-acknowledgement back-channel messages to be returned.

`ackOrNackMessageOnly` indicates that the encoder is capable of sending or the decoder is capable of receiving an H.263 bitstream which contains requests for either acknowledgement or non-acknowledgement back-channel messages to be returned, but only one for a particular video bitstream.

`ackAndNackMessage` indicates that the encoder is capable of sending or the decoder is capable of receiving an H.263 bitstream which contains requests for acknowledgement and non-acknowledgement back-channel messages to be returned.

CustomPictureClockFrequency: indicates the capability to support custom picture clock frequency when present as a capability, and parameters for custom picture clock frequency when present in `OpenLogicalChannel` and `RequestMode`.

When used in `OpenLogicalChannel`, if `customPictureClockFrequency` has more than one member in its set, then the video bitstream is allowed to switch between the various Picture Clock Frequencies (PCFs) within that set within the same video bitstream. Even if there is only one PCF in the set, if any MPI values are sent for the standard PCF at higher levels in the same message (e.g. in the same `H263VideoCapability`), then within the same bitstream there can be switching between the standard PCF and the custom one. If one wishes to indicate that the PCF should not change within the bitstream, then data relevant to only one PCF should be sent (either only MPI values for the standard PCF or just the `customPictureClockFrequency`).

`clockConversionCode` indicates the clock conversion code when custom picture clock frequency is used in H.263.

`clockDivisor` indicates the natural binary representation of the value of the clock divisor. The custom picture clock frequency is given by $1\ 800\ 000 / (\text{clock divisor} * \text{clock conversion factor})$ Hz.

If present, `sqcifMPI` indicates the minimum picture interval in units of $1/(\text{custom picture clock frequency})$ for the encoding and/or decoding of SQCIF pictures, and if not present, no capability for SQCIF pictures is indicated.

If present, `qcifMPI` indicates the minimum picture interval in units of $1/(\text{custom picture clock frequency})$ for the encoding and/or decoding of QCIF pictures, and if not present, no capability for QCIF pictures is indicated.

If present, `cifMPI` indicates the minimum picture interval in units of $1/(\text{custom picture clock frequency})$ for the encoding and/or decoding of CIF pictures, and if not present, no capability for CIF pictures is indicated.

If present, `cif4MPI` indicates the minimum picture interval in units of $1/(\text{custom picture clock frequency})$ for the encoding and/or decoding of 4CIF pictures, and if not present, no capability for 4CIF pictures is indicated.

If present, `cif16MPI` indicates the minimum picture interval in units of $1/(\text{custom picture clock frequency})$ for the encoding and/or decoding of 16CIF pictures, and if not present, no capability for 16CIF pictures is indicated.

CustomPictureFormat: indicates the capability to support a custom picture format when present as a capability, and parameters for custom picture format when present in `OpenLogicalChannel` and `RequestMode`.

The parameters `maxCustomPictureWidth`, `maxCustomPictureHeight`, `minCustomPictureWidth`, `minCustomPictureHeight` indicate the range of picture sizes that an encoder or decoder can support; and the requested picture size in the case of use with `RequestMode`.

`standardMPI` indicates the minimum picture interval in units of $1/29.97$ when no custom picture clock frequency is used.

`customPCF` indicates the parameters for custom picture clock frequency when used in conjunction with custom picture format.

`clockConversionCode` indicates a clock conversion code when custom picture clock frequency is used in H.263.

`clockDivisor` indicates the natural binary representation of the value of the clock divisor. The custom picture clock frequency is given by $1\ 800\ 000/(\text{clock divisor} * \text{clock conversion factor})$ Hz.

`customMPI` indicates the minimum picture interval in units of $1/(\text{custom picture clock frequency})$ for the encoding and/or decoding of pictures in the custom picture format size.

`pixelAspectInformation` indicates the capability of an encoder or decoder to support various pixel aspect ratios; and the requested pixel aspect ratio in the case of use with `RequestMode`.

`pixelAspectCode` indicates the capability to support the pixel aspect ratio as indicated by the PAR code of H.263.

`extendedPAR`: `width`, `height` indicate the capability to support the pixel aspect ratio as indicated by the Extended Pixel Aspect Ratio (EPAR) code of Recommendation H.263.

H263VideoModeCombos

When present, `h263VideoModeCombos` is used to indicate dependencies among optional modes of H.263. The mode combinations for which capabilities are indicated in `h263VideoModeCombos` are not implied to be allowed for use with other optional modes signalled at higher levels within the same `H263Options` or `H263VideoCapability` or `H263VideoMode` message except as noted in the fourth paragraph of this clause and in the third paragraph of the following clause. In other words, if support for some mode booleans for the same modes that contain booleans in

H263VideoModeCombos are indicated at higher levels of the syntax in h263Mode or H263Capability, these modes are not assumed to also apply in uncoupled combinations with the modes declared in H263VideoModeCombos.

h263VideoUncoupledModes indicates which optional modes of H.263 operation can be switched on or off independently to each other in any syntactically correct way for a picture and which can be switched on or off independently to the modes indicated within the h263VideoCoupledModes sent in the same H263VideoModeCombos sequence.

h263VideoCoupledModes indicates one or more sets of the optional modes of H.263 operation which can be switched on or off together for a picture within an H.263 bitstream, but for which the ability to independently switch on or off any subset of these modes is not implied. Any set of modes which are indicated as coupled in an h263VideoCoupledModes message can be used along with the full set or any subset of the modes that are indicated as uncoupled in the accompanying h263VideoUncoupledModes message within the same H263VideoModeCombos message. Within the contents of each H263ModeComboFlags message of an h263VideoCoupledModes message there shall be at least two boolean flags set to true, and there shall not be a set of mode flags set to true which indicates a coupled combination of modes which is not syntactically allowed within the same picture of an H.263 bitstream.

Some optional features of Recommendation H.263 are not included in H263ModeComboFlags since they are thought unlikely to require coupling in implementation. Specifically, these include the features specified in Annex L/H.263 (for example, fullPictureFreeze, partialPictureFreezeAndRelease, and resizingPartPicFreezeAndRelease) and the optional picture formats and optional picture clock frequencies. If support of any of these such features is signalled at a higher level within the same H263Options or H263VideoCapability or H263VideoMode message, these features shall operate in an uncoupled manner with the mode combinations signalled in H263VideoModeCombos. A fairly complex example of the use of H263VideoModeCombos follows.

The example consists of a case in which H263VideoCapability indicates that advancedPrediction and unrestrictedVector are supported, and (in the same H263VideoCapability message) inside an H263Options message it is indicated that dynamicPictureResizingByFour is supported and (in the same H263VideoCapability message) inside a H263VideoModeCombos message is an h263VideoUncoupledModes message which indicates that advancedIntraCodingMode is supported in an uncoupled manner along with an h263VideoCoupledModes message which indicates that modifiedQuantizationMode and slicesInOrder-NonRect are supported in a coupled manner. This then means that the video bitstream can contain (only) pictures with the following mode combinations: None, advancedPrediction, unrestrictedVector, dynamicPictureResizingByFour, advancedPrediction with unrestrictedVector, advancedPrediction with dynamicPictureResizingByFour, unrestrictedVector with dynamicPictureResizingByFour, advancedPrediction with unrestrictedVector with dynamicPictureResizingByFour, advancedIntraCodingMode, modifiedQuantizationMode with slicesInOrder-NonRect, and advancedIntraCodingMode with modifiedQuantizationMode with slicesInOrder-NonRect.

H263ModeComboFlags

The individual parameters of H263ModeComboFlags have the same meaning as the parameters with the same name in H263VideoCapability and H263Options.

unlimitedMotionVectors shall be FALSE if unrestrictedVector is FALSE in the same H263VideoUncoupledModes message. unlimitedMotionVectors shall be FALSE if unrestrictedVector is FALSE in the same H263VideoCoupledModes message and in the H263VideoUncoupledModes message in the same H263VideoModeCombos message.

referencePicSelect, when true, indicates the ability of the encoder or decoder to use the Reference Picture Selection mode of H.263. When true, the specific parameters specifying how the Reference Picture Selection mode can be used shall be as sent in the refPictureSelection field of the same H263Options message. referencePicSelect shall not be true unless refPictureSelection is present in the same H263Options message.

IS11172 VideoCapability: indicates IS11172 [43] capabilities.

constrainedBitstream indicates the capability for bitstreams in which constrained_parameters flag is set to "1": a value of true indicates that such operation is possible, while a value of false indicates that such operation is not possible. An encoder shall produce bitstreams within the limitations imposed by the optional fields (see below). A decoder shall be able to accept all bit streams within the limitations indicated by the optional fields. The optional fields are integers with units defined in Table B.3.

Table B.3/H.245 – Units for IS11172-2 codepoints

ASN.1 codepoint	Units for referenced parameter
videoBitRate	400 bit/s
vbvBufferSize	16 384 bits
samplesPerLine	Samples per line
linesPerFrame	Lines per frame
pictureRate	Refer to 2.4.3.2 of IS11172-2
luminanceSampleRate	Samples per second

B.2.2.6 Audio Capabilities

This indicates audio capabilities. The indication of more than a single capability within a single AudioCapability does not indicate simultaneous processing capability. Simultaneous processing capability can be indicated by instances of AudioCapability in different AlternativeCapabilitySets in a single CapabilityDescriptor.

The capability to transmit and/or receive G-series audio is indicated by a choice of integers. When an H.222.1 multiplex is used, these numbers refer to the available STD buffer size in units of 256 octets. When an H.223 multiplex is used, these numbers refer to the maximum number of audio frames per AL-SDU. When an H.225.0 multiplex is used, these numbers indicate the maximum number of audio frames per packet. The exact meaning of the codepoints is given in Table B.4.

Table B.4/H.245 – G-series audio codepoints

ASN.1 codepoint	Semantic meaning of codepoint
g711Alaw64k	G.711 audio at 64 kbit/s, A-law
g711Alaw56k	G.711 audio at 56 kbit/s, A-law, truncated to 7 bits
g711Ulaw64k	G.711 audio at 64 kbit/s, μ -law
g711Ulaw56k	G.711 audio at 56 kbit/s, μ -law, truncated to 7 bits
g722-64k	G.722 7 kHz audio at 64 kbit/s
g722-56k	G.722 7 kHz audio at 56 kbit/s
g722-48k	G.722 7 kHz audio at 48 kbit/s

Table B.4/H.245 – G-series audio codepoints (concluded)

ASN.1 codepoint	Semantic meaning of codepoint
g7231	G.723.1 at either 5.3 or 6.3 kbit/s
g728	G.728 audio at 16 kbit/s
g729	G.729 audio at 8 kbit/s
g729AnnexA	G.729 Annex A audio at 8 kbit/s
g729wAnnexB	G.729 audio at 8 kbit/s with silence suppression as in Annex B
g729AnnexAwAnnexB	G.729 Annex A audio at 8 kbit/s with silence suppression as in Annex B
g7231AnnexCCapability	G.723.1 with Annex C
gsmFullRate	Full rate speech transcoding (GSM 06.10)
gsmHalfRate	Half rate speech transcoding (GSM 06.20)
gsmEnhancedFullRate	Enhanced Full Rate (EFR) speech transcoding (GSM 06.60)

g7231: indicates the ability to process audio codec G723.1. `maxAl-sduAudioFrames` indicates the maximum number of audio frames per AL-SDU. The boolean `silenceSuppression`, when true, indicates the capability to use silence compression defined in Annex A/G.723.1.

g7231AnnexCCapability: indicates the ability to process audio codec G723.1 with its Annex C. `maxAl-sduAudioFrames` indicates the maximum number of audio frames per AL-SDU. The boolean `silenceSuppression`, when true, indicates the capability to use silence compression defined in Annex A/G.723.1. `g723AnnexCAudioMode` shall not be present when `G7231AnnexCCapability` is included in a `TerminalCapabilitySet` message, but shall be present when `G7231AnnexCCapability` is included in an `OpenLogicalChannel` message. The fields `highRateMode0`, `highRateMode1`, `lowRateMode0`, `lowRateMode1`, `sidMode0`, and `sidMode1` indicate the number of octets per frame for each of the audio and error protection modes of G.723.1 and Annex C/G.723.1 that will be used on the logical channel.

IS11172AudioCapability: indicates the ability to process audio coded according to ISO/IEC 11172-3 [44].

Booleans that have the value of true indicate that the particular mode of operation is possible, while a value of false indicates that it is not. The booleans `audioLayer1`, `audioLayer2` and `audioLayer3` indicate which audio coding layers can be processed. The booleans `audioSampling32k`, `audioSampling44k1` and `audioSampling48k` indicate which of the audio sample rates, 32 kHz, 44.1 kHz and 48 kHz respectively, can be processed. The booleans `singleChannel` and `twoChannels` indicate capability for single channel and stereo/dual channel operation respectively. The integer `bitRate` indicates the maximum audio bit rate capability, and is measured in units of kbit/s.

IS13818AudioCapability: indicates the ability to process audio coded according to ISO/IEC 13818-3 [45].

Booleans that have the value of true indicate that the particular mode of operation is possible, while a value of false indicates that it is not. The booleans `audioLayer1`, `audioLayer2` and `audioLayer3` indicate which audio coding layers can be processed. The booleans `audioSampling16k`, `audioSampling22k05`, `audioSampling24k`, `audioSampling32k`, `audioSampling44k1` and `audioSampling48k` indicate which of the audio sample rates, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz and 48 kHz respectively, can be processed.

The booleans concerned with multi-channel operation indicate capability to operate in the particular modes, as specified in Table B.5.

Table B.5/H.245 – ISO/IEC 13818-3 multi-channel codepoints

ASN.1 codepoint	Semantic meaning of codepoint
singleChannel	One channel, using the 1/0 configuration. Single channel mode (as in ISO/IEC 11172-3).
twoChannels	Two channels, using the 2/0 configuration. Stereo or dual channel mode (as in ISO/IEC 11172-3).
threeChannels2-1	Three channels, using the 2/1 configuration. Left, Right and single surround channel.
threeChannels3-0	Three channels, using the 3/0 configuration. Left, Centre and Right, without surround channel.
fourChannels2-0-2-0	Four channels, using the 2/0 + 2/0 configuration. Left and Right of the first programme and Left and Right of the second programme.
fourChannels2-2	Four channels, using the 2/2 configuration. Left, Right, Left surround and Right surround.
fourChannels3-1	Four channels, using the 3/1 configuration. Left, Centre, Right, and a single surround channel.
fiveChannels3-0-2-0	Five channels, using the 3/0 + 2/0 configuration. Left, Centre and Right of the first programme and Left and Right of the second programme.
fiveChannels3-2	Five channels, using the 3/2 configuration. Left, Centre, Right, Left surround and Right surround.

The boolean lowFrequencyEnhancement indicates the capability for a low frequency enhancement channel.

The boolean multilingual, when true, indicates the capability to support up to seven multilingual channels, and when false that no multilingual channel is supported.

The integer bitRate indicates the maximum audio bit rate capability, and is measured in units of kbit/s.

GSMAudioCapability: indicates capabilities for the GSM full rate, half rate and enhanced full rate speech transcoding audio codecs. audioUnitSize indicates the maximum number of bytes to be sent in each packet, comfortNoise, when true, indicates the capability to support the comfort noise processing for the full, half or enhanced full rate speech traffic channel (GSM 06.12, GSM 06.22, GSM 06.62), and scrambled, when true, indicates the capability to support bit scrambling for the full, half, or enhanced full rate speech traffic channels (GSM 06.10, GSM 06.20, GSM 06.60).

B.2.2.7 Data Application Capabilities

This indicates data capabilities. The indication of more than a single capability within a single DataApplicationCapability does not indicate simultaneous processing capability. Simultaneous processing capability can be indicated by instances of DataApplicationCapability in different AlternativeCapabilitySets in a single CapabilityDescriptor.

Recommendations that use this Recommendation may place restrictions on which of these modes may be signalled.

Some of the data capabilities require bidirectional logical channels, for example, to run a retransmission protocol. This requirement is implicitly included in the appropriate capability codepoints.

DataApplicationCapability: is a list of data applications and bit rates. Each data application indicated shall be supported by one or more DataProtocolCapabilities.

maxBitRate indicates the maximum bit rate in units of 100 bit/s at which a transmitter can transmit video or a receiver can receive the given data application.

t120 indicates the capability to support the T.120 [32] protocol.

dsm-cc indicates the capability to support the DSM-CC [46] protocol.

userData indicates the capability to support unspecified user data from external data ports.

t84 indicates the capability to support the transfer of T.84 [31] type images (JPEG, JBIG, Facsimile Gr.3/4).

t434 indicates the capability to support the transfer of T.434 [34] telematic binary files.

h224 indicates the capability to support the real-time simplex device control protocol H.224 [11].

nlpid indicates the capability to support the network layer protocol as specified by nlpidData as defined in ISO/IEC TR 9577 [47]. These protocols include Internet Protocol (IP) and IETF Point-to-Point Protocol (PPP), among others.

NOTE – The use of the NLPID is extensively described in IETC RFC1490 "Multiprotocol Interconnect over Frame Relay".

dsvdControl indicates the capability of the DSVD terminal to support an out-of-band control channel.

h222DataPartitioning indicates the capability to support the modified and restricted usage of data partitioning of H.262, as specified in Recommendation H.222.1, in which the enhancement data is transmitted as a data channel supported by the listed DataProtocolCapability.

t30fax: this codepoint indicates the capability to use Annex C/T.30 analog mode (G3V), as specified in Recommendation T.39 for the DSVF/MSVF modes.

t140: this codepoint indicates the capability to use the T.140 text conversation protocol, as specified in Recommendation T.140.

t38fax: this codepoint indicates a data protocol conforming to Recommendation T.38 [29].

fillBitRemoval, when true, indicates that the gateway/terminal has the ability to remove and insert fill bits.

transcodingJBIG, when true, indicates that the gateway has the ability to transcode in real time between the line compression and JBIG for transfer over the IP network.

transcodingMMG, when true, indicates that the gateway has the ability to transcode in real time between the line compression and MMG for transfer over the IP network.

DataProtocolCapability: contains a list of data protocols.

v14buffered indicates the capability to support a specified data application using buffered V.14 [35].

v42lapm indicates the capability to support a specified data application using the LAPM protocol defined in Recommendation V.42 [37].

hdlcFrameTunnelling indicates the capability to support a specified data application using HDLC Frame Tunnelling. Refer to 4.5.2 of ISO/IEC 3309 [42].

h310SeparateVCStack indicates the capability to support a specified data application using the protocol stack defined in Recommendation H.310 for the transport of H.245 messages over a separate ATM VC to that used for audiovisual communication.

h310SingleVCStack indicates the capability to support a specified data application using the protocol stack defined in Recommendation H.310 for the transport of H.245 messages in the same ATM VC as that used for audiovisual communication.

transparent indicates the capability to support a specified data application using transparent data transfer.

v120: use of v120 is for further study in Recommendation H.323.

separateLANStack indicates that a separate transport stack will be used to transport the data. The intent of a separate network connection for data is indicated by dataType in OpenLogicalChannel resolving to values h310SeparateVCStack or separateLANStack of DataProtocolCapability. When the selected DataApplicationCapability is t120, these choices imply use of the T.123 basic profile for B-ISDN and LAN, respectively. Alternative LAN profiles may be selected by a nonStandard DataProtocolCapability.

If separateLANStack is selected and separateStack is present in the OpenLogicalChannel request, the receiver should attempt to establish the stack indicated. It will respond OpenLogicalChannelAck if successful, otherwise OpenLogicalChannelReject with a suitable cause.

If separateLANStack is selected and separateStack is absent in the OpenLogicalChannel request, the receiver should supply an appropriate separateStack in its OpenLogicalChannelAck response. The receiver of this (the original requester) should then attempt to establish the stack indicated. It will issue CloseLogicalChannel if unsuccessful.

If separateLANStack is selected and separateStack is present in the OpenLogicalChannel request, it can be overridden by separateStack in the OpenLogicalChannelAck response. If the original requester does not tolerate an override, it will issue CloseLogicalChannel.

If separateLANStack is selected and separateStack is absent in the OpenLogicalChannel request and also absent in the OpenLogicalChannelAck response, the original requester can infer that the responder does not understand these ASN.1 extensions and should issue CloseLogicalChannel to clean up.

v76wCompression indicates the capability to support data compression on a V.76 data channel.

tcp indicates the capability to support TCP/IP for this application.

udp indicates the capability to support UDP for this application.

T84Profile: indicates the types of still image profile that the terminal is able to support.

t84Unrestricted provides no indication of the type of T.84 still image that the terminal is able to support: information in the T.84 layer should be used to determine whether a particular image can be received.

t84Restricted indicates the type of T.84 still image that the terminal is able to support.

qcif indicates the support of a sequential colour YCrCb type image with QCIF resolution.

cif indicates the support of a sequential colour YCrCb type image with CIF resolution.

ccir601Seq indicates the support of a sequential colour YCrCb type image with CCIR601 resolution.

ccir601Prog indicates the support of a progressive colour YCrCb type image with CCIR601 resolution.

hdtvSeq indicates the support of a sequential colour YCrCb type image with HDTV resolution.

hdtvProg indicates the support of a progressive colour YCrCb type image with HDTV resolution.

g3FacsmH200x100 indicates the support of a sequential Facsimile Gr. 3 MH (Modified Huffman) coded bi-level image at the normal (200×100ppi) resolution.

g3FacsmH200x200 indicates the support of a sequential Facsimile Gr. 3 MH (Modified Huffman) coded bi-level image at the high (200×200ppi) resolution.

g4FacsmMR200x100 indicates the support of a sequential Facsimile Gr. 4 MMR (Modified Modified Reed) coded bi-level image at the normal (200×100ppi) resolution.

g4FacsmMR200x200 indicates the support of a sequential Facsimile Gr. 4 MMR (Modified Modified Reed) coded bi-level image at the high (200×200ppi) resolution.

jbig200x200Seq indicates the support of a sequential bi-level JBIG coded bi-level image at the 200×200ppi resolution.

jbig200x200Prog indicates the support of a progressive bi-level JBIG coded bi-level image at the 200×200ppi resolution.

jbig300x300Seq indicates the support of a sequential bi-level JBIG coded bi-level image at the 300×300ppi resolution.

jbig300x300Prog indicates the support of a progressive bi-level JBIG coded bi-level image at the 300×300ppi resolution.

digPhotoLow indicates the support of a sequential JPEG coded colour image of up to 720×576 image size.

digPhotoMedSeq indicates the support of a sequential JPEG coded colour image of up to 1440×1152 image size.

digPhotoMedProg indicates the support of a progressive JPEG coded colour image of up to 1440×1152 image size.

digPhotoHighSeq indicates the support of a sequential JPEG coded colour image of up to 2880×2304 image size.

digPhotoHighProg indicates the support of a progressive JPEG coded colour image of up to 2880×2304 image size.

B.2.2.8 Encryption, Authentication and Integrity Capabilities

EncryptionCapability, if present, indicate the encryption capabilities of a terminal for each media type where the capabilities are present. The scope of encryption indicates whether the encryption is applied to the entire bit stream, a portion of the bit stream in a standard way, or a portion of the stream in a non-standard way. The algorithm selects the encryption algorithm.

AuthenticationCapabilities if present, indicate that the authentication components of H.235 [16] are supported by the terminal.

IntegrityCapabilities if present, indicate that the integrity components of H.235 [16] are supported by the terminal.

B.2.2.9 Conference Capabilities

ConferenceCapability indicates conference capabilities such as the ability to support Chair Control as described in Recommendation H.243.

B.2.2.10 User Input Capabilities

UserInputCapability indicates which parameters in the UserInputIndication message are supported by the terminal. basicString indicates that the terminal supports the basicString option of userInputSupportIndication, iA5String indicates that the terminal supports the iA5String option of userInputSupportIndication and generalString indicates that the terminal supports the generalString option of userInputSupportIndication. dtmf indicates that the terminal supports dtmf using the signal and signal update components of the userInputIndication message. hookflash indicates that the terminal supports hookflash using the signal and signal update components of the userInputIndication message.

B.2.3 Terminal Capability Set Acknowledge

This is used to confirm receipt of a TerminalCapabilitySet from the peer CESE.

The sequenceNumber shall be the same as the sequenceNumber in the TerminalCapabilitySet for which this is the confirmation.

B.2.4 Terminal Capability Set Reject

This is used to reject a TerminalCapabilitySet from the peer CESE.

The sequenceNumber shall be the same as the sequenceNumber in the TerminalCapabilitySet for which this is the negative acknowledgement.

The reasons for sending this message are given in Table B.6.

Table B.6/H.245 – Reasons for rejecting a TerminalCapabilitySet

ASN.1 codepoint	Cause
unspecified	No cause for rejection specified.
UndefinedTableEntryUsed	A capability descriptor made reference to a capabilityTable entry that is not defined.
DescriptorCapacityExceeded	The terminal was incapable of storing all of the information in the TerminalCapabilitySet.
TableEntryCapacityExceeded	The terminal was incapable of storing more entries than that indicated in highestEntryNumberProcessed or else could not store any.

B.2.5 Terminal Capability Set Release

This is sent in the case of a timeout.

B.3 Logical channel signalling messages

This set of messages is for logical channel signalling. The same set of messages is used for unidirectional and bidirectional logical channel signalling; however, some parameters are only present in the case of bidirectional logical channel signalling.

"Forward" is used to refer to transmission in the direction from the terminal making the original request for a logical channel to the other terminal, and "reverse" is used to refer to the opposite direction of transmission, in the case of a bidirectional channel request.

B.3.1 Open Logical Channel

This is used to attempt to open a unidirectional logical channel connection between an outgoing LCSE and a peer incoming LCSE and to open a bidirectional logical channel connection between an outgoing B-LCSE and a peer incoming B-LCSE.

forwardLogicalChannelNumber: indicates the logical channel number of the forward logical channel that is to be opened.

forwardLogicalChannelParameters: include parameters associated with the logical channel in the case of attempting to open a unidirectional channel and parameters associated with the forward logical channel in the case of attempting to open a bidirectional channel.

reverseLogicalChannelParameters: include parameters associated with the reverse logical channel in the case of attempting to open a bidirectional channel. Its presence indicates that the request is for a bidirectional logical channel with the stated parameters, and its absence indicates that the request is for a unidirectional logical channel.

NOTE – H.222 parameters are not included in reverseLogicalChannelParameters as their values are not known to the terminal initiating the request.

portNumber is a user-to-user parameter that may be used by a user for such purposes as associating an input or output port, or higher layer channel number, with the logical channel.

dataType indicates the data that is to be carried on the logical channel.

If it is nullData, the logical channel will not be used for the transport of elementary stream data, but only for adaptation layer information – if video is to be transmitted in one direction only, but a retransmission protocol is to be used, such as AL3 defined in Recommendation H.223, a return channel is needed to transport the retransmission requests – it may also be used to describe a logical channel that only contains PCR values in the case of H.222.1 Transport Streams [9].

Terminals capable only of unidirectional (transmit or receive) operation on media types which make use of bidirectional channels shall send capabilities only for the supported direction of operation. The reverse direction shall use the nullData type, for which no capability is necessary. Transmit-only terminals should send transmit capabilities, but terminals should not assume that the absence of transmit capabilities implies that transmit-only operation is not possible.

separateStack indicates that a separate transport stack will be used to transport the data and provides an address to use to establish the stack which is either a Q.2931, E.164, or local area network transport address.

networkAccessParameters define the distribution, network address, and creation and association information to be used for the separateStack.

distribution shall be present when networkAddress is set to localAreaNetwork and shall indicate whether the networkAddress is a uni or multicast transport address.

networkAddress indicates the address of the actual stack in use: Q.2931, E.164, or local area network transport address.

associateConference indicates whether or not the data conference is new (associateConference = FALSE) or is an existing data conference which should be associated with the audio/video call (associateConference = TRUE).

externalReference indicates information which may be used to further provide association or information concerning the separateStack.

If it is of type VideoCapability, AudioCapability, the logical channel may be used for any of the variations indicated by each individual capability; and it shall be possible to switch between these

variations using only signalling that is in-band to the logical channel – for example, in the case of H.261 video, if both QCIF and CIF are indicated, it shall be possible to switch between these on a picture-by-picture basis. In the case of DataApplicationCapability, only one instance of a capability can be indicated since there is no in-band signalling allowing a switch between variations.

If it is encryptionData, the logical channel will be used for the transport of encryption information as specified.

forwardLogicalChannelDependency indicates which logical channel number the forward channel to be opened is dependent on.

reverseLogicalChannelDependency indicates which logical channel number the reverse channel to be opened is dependent on.

The replacementFor parameter indicates that the logical channel to be opened will be a *replacement for* the specified existing, already-open logical channel. This parameter shall be used only to refer to logical channels already in the ESTABLISHED state. Logical channels opened using this parameter shall not carry any data traffic until after all traffic on the referenced established logical channel ceases. Media decoders will in this case never be required to decode data traffic from both logical channels simultaneously. Once traffic on the newly established logical channel has begun, the old logical channel shall immediately be closed. Receivers may acknowledge logical channels opened using the replacementFor mechanism with the understanding that the old and new logical channels shall not be used simultaneously, and therefore will not exceed the receiver's capability to decode.

The encryptionSync field shall be used by the master when acknowledging the opening of a channel by a slave. It is used to provide the encryption key value and the synchronization point at which the key should be used. For H.323, the syncFlag shall be set to the RTP dynamic payload number which matches the key.

H222LogicalChannelParameters: is used to indicate parameters specific to using H.222.1 [9]. It shall be present in forwardLogicalChannelParameters and shall not be present in reverseLogicalChannelParameters.

resourceID indicates in which ATM Virtual Channel the logical channel is to be transported. The means by which this parameter is associated with an ATM Virtual Channel is not specified in this Recommendation.

subChannelID indicates which H.222.1 subchannel is used for the logical channel. It shall be equal to the PID in a Transport Stream and the stream_id in a Program Stream.

pcr-pid indicates the PID used for the transport of Program Clock References when the Transport Stream is used. It shall be present when the ATM virtual channel carries a Transport Stream and shall not be present when the ATM virtual channel carries a Program Stream.

programDescriptors is an optional octet string, which, if present, contains one or more descriptors, as specified in Recommendations H.222.0 and H.222.1, that describe the program that the information to be carried in the logical channel is a part of.

streamDescriptors is an optional octet string, which, if present, contains one or more descriptors, as specified in Recommendations H.222.0 and H.222.1, that describe the information that is to be carried in the logical channel.

H223LogicalChannelParameters: is used to indicate parameters specific to using H.223 [10]. It shall be present in forwardLogicalChannelParameters and reverseLogicalChannelParameters.

adaptationLayerType indicates which adaptation layer and options will be used on the logical channel. The codepoints are as follows: nonStandard, allFramed (AL1 framed mode), allNotFramed (AL1 unframed mode), al2WithoutSequenceNumbers (AL2 with no sequence numbers present),

al2WithSequenceNumbers (AL2 with sequence numbers present), and al3 (AL3, indicating the number of control field octets that will be present and the size of the send buffer, B_S , that will be used, the size being measured in octets), al1M (AL1M defined in Annex C with the specified parameters), al2M (AL2M defined in Annex C with the specified parameters) or al3M (AL3M defined in Annex C with the specified parameters).

segmentableFlag, when equal to true indicates that the channel is designated to be segmentable, and when equal to false indicates that the channel is designated to be non-segmentable.

H223AL1MParameters: is used to indicate parameters specific to using adaptation Layer AL1M.

transferMode indicates whether framed mode or unframed mode is used.

headerFEC indicates whether FEC is SEBCH(16,7) or Golay(24,12).

The length of CRC bits for the payload is indicated by crcLength as 4, 12, 20 or 28 bits.

rcpcCodeRate indicates the RCPC code rate as 8/8, 8/9, ..., 8/32.

arqType indicates the ARQ mode of operation: noARQ indicates no retransmission, typeIArq indicates ARQ type I, and typeIIArq indicates ARQ type II.

alpduInterleaving, if true, indicates the use of AL-PDU interleaving.

alsduSplitting, if true, indicates the use of AL-SDU splitting mode.

H223AL2MParameters: is used to indicate parameters specific to using adaptation Layer AL2M.

headerFEC indicates whether FEC is SEBCH(16,5) or Golay(24,12).

alpduInterleaving, if true, indicates the use of AL-PDU interleaving.

H223AL3MParameters: is used to indicate parameters specific to using adaptation Layer AL3M.

This has the same parameters as AL1MParameters, except transferMode and alsduSplitting are not present.

H223AnnexCArqParameters

numberOfRetransmissions indicates the maximum number of retransmissions that may be used: finite indicates a finite limit on the number of retransmissions that may be used in the range 0 to 16; and infinite indicates that there is no limit on the number of retransmissions that may be used; numberOfRetransmissions equal to the finite value of 0 indicates that the control field is used for the splitting mode but retransmissions are not used.

sendBufferSize indicates the size of the send buffer that will be used, the size being measured in octets.

V76LogicalChannelParameters: is used to indicate parameters specific to using V.76.

audioHeader is used to indicate the use of an audio header on the logical channel. This is a valid parameter for channels of the DataType audio.

suspendResume is used to indicate that the channel may use the suspend/resume procedures to suspend other logical channels. Three channel options may be selected; no suspend resume on the channel, suspend resume using an address or suspend resume without an address as defined in Recommendation V.76. suspendResumewAddress indicates that the suspend/resume channel shall use the address field as defined in Recommendation V.76. suspendResumewoAddress indicates that the suspend/resume channel shall not use the address field.

eRM indicates that the logical channel shall perform error recovery procedures as defined in Recommendation V.76.

uNERM indicates that the logical channel shall operate in non-error recovery mode as defined in Recommendation V.76.

For description of n401, windowSize and loopbackTestProcedure see 12.2.1/V.42, and its subclauses. For the purposes of V.70, n401 shall be encoded in octets.

crcLength is an optional parameter that indicates the CRC length used in error recovery mode. If this parameter is not present, the default CRC length shall be used. crc8bit indicates to use an 8-bit CRC, crc16bit indicates use of the 16-bit CRC and crc32bit indicates to use a 32-bit CRC as defined in Recommendation V.76.

recovery is an optional parameter that indicates the error recover procedures defined in Recommendation V.76. If this parameter is not present, the default error recovery procedure shall be used. sREJ indicates to use the selective frame reject procedure and mSREJ indicates to use the multiple selective reject procedure as defined in Recommendation V.76.

uIH indicates the use of V.76 UIH frames.

rej indicates the use of the reject procedure in V.76.

V75Parameters is used to indicate parameter specific to using V.75. audioHeaderPresent indicates the presence of the V.75 audio header.

H2250LogicalChannelParameters: is used to indicate parameters specific to using H.225.0. It shall be present in forwardLogicalChannelParameters and reverseLogicalChannelParameters.

The sessionID is a unique RTP or T.120 Session Identifier in the conference. It is used by the transmitter to refer to the session to which the logical channel applies. Only the master can create the session identification. By convention, there are three primary sessions. The first primary session with a session identification of 1 is the audio session, the second primary session with a session identification of 2 is the video session, and the third primary session with a session identification of 3 is the data session. A slave entity can open an additional session by providing a session identification of 0 in the openLogicalChannel message. The master will create a unique session identification and provide it in the openLogicalChannelAck message.

The associatedSessionID is used to associate one session with another. Typical use will be to associate an audio session with a video session to indicate which sessions to process for lip synchronization.

The mediaChannel indicates a transportAddress to be used for the logical channel. It is not present in the OpenLogicalChannel message when the transport is unicast. If the transportAddress is multicast, the master is responsible for creating the multicast transport address and shall include the address in the OpenLogicalChannel message. A slave entity that wishes to open a new multicast channel will provide zeroes in the multicast transportAddress field. The master will create and provide the multicast transportAddress in the OpenLogicalChannelAck message for the slave entity. Note that the MC will use the communicationModeCommand to specify the details about all the RTP Sessions in the conference.

The mediaChannel is used to describe the transport address for the logical channel. IPv4 and IPv6 addresses shall be encoded with the most significant octet of the address being the first octet in the respective OCTET STRING, e.g. the class B IPv4 address 130.1.2.97 shall have the "130" being encoded in the first octet of the OCTET STRING, followed by the "1" and so forth. The IPv6 address a148:2:3:4:a:b:c:d shall have the "a1" encoded in the first octet, "48" in the second, "00" in the third, "02" in the fourth and so forth. IPX addresses, node, netnum, and port shall be encoded with the most significant octet of each field being the first octet in the respective OCTET STRING.

mediaGuaranteedDelivery indicates whether or not the underlying media transport should be selected to provide or not provide guaranteed delivery of data.

mediaControlChannel indicates the media control channel in which the sender of the open logical channel will be listening for media control messages for this session. This field is present only when a media control channel is required.

mediaControlGuaranteedDelivery indicates whether or not the underlying media control transport should be selected to provide or not provide guaranteed delivery of data. This field is present only when a media control channel is required.

The silenceSuppression is used to indicate whether the transmitter stops sending packets during times of silence. It shall be included in the openLogicalChannel message for an audio channel and omitted for any other type of channel.

destination indicates the terminalLabel of the destination if one has been assigned.

dynamicRTPPayloadType indicates a dynamic payload value which is used in H.323 for the H.225.0 alternative H.261 video packetization scheme. This field is present only when a dynamic RTP payload is in use.

mediaPacketization indicates which optional media packetization scheme is in use.

redundancyEncoding indicates that the redundant encoding method indicated in this parameter is to be used for the logical channel to be opened. The primary encoding is defined by the *dataType* of the *forwardLogicalChannelParameters* or the *reverseLogicalChannelParameters*, respectively. The type of redundancy encoding to be applied for this logical channel is identified by the *redundancyEncodingMethod* parameter, the secondary encoding is specified in the *secondaryEncoding* parameter. The *DataType* (audio, video, etc.) selected for both primary and secondary encoding shall match and shall be in accordance with the *redundancyEncodingMethod* selected.

The source parameter is used to identify the terminal number of the sender of the OpenLogicalChannel message.

h235 Key: is used to include, and specify the method by which media specific session keys are protected as they are passed between two endpoints. The encoding of this field is a nested ASN.1 value as described in Recommendation H.235.

EscrowData is used to specify the type and contents of any key escrow mechanism in use. Specific types and contents may be required by implementations when media encryption is enabled.

T120SetupProcedure indicates how the T.120 conference is to be set up. For originateCall and waitForCall, the caller should derive the T.120 numeric conference name from the H.323 CID (as described in Recommendation H.323), and issue the appropriate PDU (if the endpoint is master, it should issue an invite request, while a slave should issue a join request). For issueQuery, the caller should first issue a query request, and then set up the T.120 conference in accordance with the contents of the query response (as described in Recommendation T.124).

B.3.2 Open Logical Channel Acknowledge

This is used to confirm acceptance of the logical channel connection request from the peer LCSE or B-LCSE. In the case of a request for a unidirectional logical channel, it indicates acceptance of that unidirectional logical channel. In the case of a request for a bidirectional logical channel, it indicates acceptance of that bidirectional logical channel, and indicates the appropriate parameters of the reverse channel.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel that is being opened.

reverseLogicalChannelParameters is present if and only if responding to a bidirectional channel request.

reverseLogicalChannelNumber indicates the logical channel number of the reverse channel.

portNumber is a user-to-user parameter that may be used by a user for such purposes as associating an input or output port, or higher layer channel number, with the reverse logical channel.

multiplexParameters indicate parameters specific to the multiplex, H.222, H.223, or H.225.0, that is used to transport the reverse logical channel.

FlowControlToZero indicates whether the transmitter is allowed to start transmitting on the logical channel. If set to true, it indicates that the transmitter should not transmit on the logical channel until receiving a subsequent FlowControl message, applying to the logical channel, allowing it to do so. If set to false, or absent, the transmitter is allowed to begin transmitting immediately the channel is established.

The replacementFor parameter indicates that the logical channel to be opened will be a *replacement for* the specified existing, already-open logical channel. This parameter shall be used only to refer to logical channels already in the ESTABLISHED state. Logical channels opened using this parameter shall not carry any data traffic until after all traffic on the referenced established logical channel ceases. Media decoders will in this case never be required to decode data traffic from both logical channels simultaneously. Once traffic on the newly established logical channel has begun, the old logical channel shall immediately be closed. Receivers may acknowledge logical channels opened using the replacementFor mechanism with the understanding that the old and new logical channels shall not be used simultaneously, and therefore will not exceed the receiver's capability to decode.

separateStack indicates that a separate transport stack will be used to transport the data and provides an address, to use to establish the stack, which is either a Q.2931, E.164, or local area network transport address.

forwardMultiplexAckParameters indicate parameters specific to the multiplex, H.222, H.223, or H.225.0 that is used to transport the forward logical channel.

The encryptionSync field shall be used by the master when acknowledging the opening of a channel by a slave. It is used to provide the encryption key value and the synchronization point at which the key should be used. For H.323, the syncFlag shall be set to the RTP dynamic payload number which matches the key.

H2250LogicalChannelAckParameters are used to indicate parameters specific to using H.225.0.

sessionID is a unique RTP Session Identifier in the conference that can only be created by the master. It is created and provided by the master if the slave wishes to create a new session by specifying an invalid session identification of 0 in the openLogicalChannelAck message.

The mediaChannel indicates a transportAddress to be used for the logical channel. It shall be present in the OpenLogicalChannelAck message when the transport is unicast. If the transportAddress is multicast, the master is responsible for creating the multicast transport address and shall include the address in the OpenLogicalChannel message. A slave entity that wishes to open a new multicast channel will provide zeroes in the multicast transportAddress field. The master will create and provide the multicast transportAddress in the OpenLogicalChannelAck message for the slave entity. Note that the MC will use the communicationModeCommand to specify the details about all the RTP Sessions in the conference.

The mediaChannel is used to describe the transport address for the logical channel. IPv4 and IPv6 addresses shall be encoded with the most significant octet of the address being the first octet in the respective OCTET STRING, e.g. the class B IPv4 address 130.1.2.97 shall have the "130" being encoded in the first octet of the OCTET STRING, followed by the "1" and so forth. The IPv6 address a148:2:3:4:a:b:c:d shall have the "a1" encoded in the first octet, "48" in the second, "00" in the third, "02" in the fourth and so forth. IPX addresses, node, netnum, and port shall be encoded with the most significant octet of each field being the first octet in the respective OCTET STRING.

mediaControlChannel indicates the media control channel in which the sender of the openLogicalChannelAck will be listening for media control messages for this session. This field is present only when a media control channel is required.

dynamicRTPPayloadType indicates a dynamic payload value which is used in H.323 for the H.225.0 alternative H.261 video packetization scheme. This field is present only when a dynamic RTP payload is in use.

The portNumber field is used in Annex C/H.323 when the receiving endpoint finds that the B-HLI given by the portNumber field in the OpenLogicalChannel message is inappropriate, and indicates the alternative value that is to be used.

NOTE – H.223 parameters are not included in reverseLogicalChannelParameters as their values were specified in the OpenLogicalChannel request message.

B.3.3 Open Logical Channel Reject

This is used to reject the logical channel connection request from the peer LCSE or B-LCSE.

NOTE – In the case of a bidirectional channel request, rejection applies to both forward and reverse channels. It is not possible to accept one and reject the other.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel specified in the request that is being rejected.

The cause field indicates the reason for rejection of the logical channel establishment. The cause values are given in Table B.7.

Table B.7/H.245 – Reasons for rejecting an OpenLogicalChannel

ASN.1 codepoint	Cause
unspecified	No cause for rejection specified.
unsuitableReverseParameters	This shall only be used to reject a bidirectional logical channel request when the only reason for rejection is that the requested reverseLogicalChannelParameters are inappropriate. Such a rejection shall immediately be followed by initiating procedures to open a similar but acceptable bidirectional logical channel.
dataTypeNotSupported	The terminal was not capable of supporting the dataType indicated in OpenLogicalChannel.
dataTypeNotAvailable	The terminal was not capable of supporting the dataType indicated in OpenLogicalChannel simultaneously with the dataTypes of logical channels that are already open.
unknownDataType	The terminal did not understand the dataType indicated in OpenLogicalChannel.

Table B.7/H.245 – Reasons for rejecting a OpenLogicalChannel (concluded)

ASN.1 codepoint	Cause
dataTypeALCombinationNotSupported	The terminal was not capable of supporting the dataType indicated in OpenLogicalChannel simultaneously with the Adaptation Layer type indicated in H223LogicalChannelParameters.
multicastChannelNotAllowed	Multicast Channel could not be opened.
insufficientBandwidth	The channel could not be opened because permission to use the requested bandwidth for the logical channel was denied.
separateStackEstablishmentFailed	A request to run the data portion of a call on a separate stack failed.
invalidSessionID	Attempt by slave to set SessionID when opening a logical channel to the master.
masterSlaveConflict	Attempt by slave to open logical channel in which the master has determined a conflict may occur. (See 8.4.1.3 and 8.5.1.3.)
waitForCommunicationMode	Attempt to open logical channel before MC has transmitted the CommunicationModeCommand.
invalidDependentChannel	Attempt to open logical channel with a dependent channel specified which is not present.
replacementForRejected	A logical channel of the type attempted cannot be opened using the replacementFor parameter. The transmitter may wish to retry by firstly closing the logical channel which was to be replaced, and then opening the replacement.

B.3.4 Open Logical Channel Confirm

This is used in bidirectional signalling to indicate to the incoming B-LCSE that the reverse channel is open and can be used for transmission.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel which was opened.

B.3.5 Close Logical Channel

This is used by the outgoing LCSE or B-LCSE to close a logical channel connection between two peer LCSEs or B-LCSEs.

NOTE – In the case of a bidirectional logical channel, this closes both forward and reverse channels. It is not possible to close one and not the other.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that is to be closed.

The source of the logical channel release is given in Table B.8.

reason indicates why the channel is being closed. reservationFailure indicates that a QoS reservation was not able to be placed for the channel and it is therefore being closed. reopen indicates that the endpoint should close the channel and then reopen a channel using the OpenLogicalChannel procedures. As an example, this may occur if a multipoint call is reduced to a point-to-point call due to endpoints dropping from a conference.

Table B.8/H.245 – Sources of logical channel release

ASN.1 codepoint	Cause
User	The LCSE or B-LCSE user is the source of the release.
Lcse	The LCSE or B-LCSE is the source of the release. This may occur as a result of a protocol error.

B.3.6 Close Logical Channel Acknowledge

This is used to confirm the closing of a logical channel connection.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that is being closed.

B.3.7 Request Channel Close

This is used to by the outgoing CLCSE to request the closing of a logical channel connection between two peer LCSEs.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that is requested to close.

qosCapability is used to indicate the QoS parameters which were in use on the channel.

reason indicates why the request to close the channel is occurring. reservationFailure indicates that a QoS reservation was not able to be placed for the channel and it is therefore being closed. reopen indicates that the endpoint should close the channel and then reopen a channel using the OpenLogicalChannel procedures. As an example, this may occur if a multipoint call is reduced to a point-to-point call due to endpoints dropping from a conference.

B.3.8 Request Channel Close Acknowledge

This is used by the incoming CLCSE to indicate that the logical channel connection will be closed.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that it has been requested to close.

B.3.9 Request Channel Close Reject

This is used by the incoming CLCSE to indicate that the logical channel connection will not be closed.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that it has been requested to close.

The cause field indicates the reason for rejection of the request to close the logical channel. The only valid cause value is unspecified.

B.3.10 Request Channel Close Release

This is sent by the outgoing CLCSE in the case of a timeout.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel of the logical channel that it has requested to close.

B.4 Multiplex Table signalling messages

This set of messages is for the secure transmission of H.223 multiplex table entries from the transmitter to the receiver.

B.4.1 Multiplex Entry Send

This is used to send H.223 multiplex table entries from the transmitter to the receiver. It is sent from an outgoing MTSE and a peer incoming MTSE.

sequenceNumber is used to label instances of MultiplexEntrySend so that the corresponding response can be identified.

MultiplexEntryDescriptors is a set of 1 to 15 MultiplexEntryDescriptors.

MultiplexEntryDescriptor: describes a single multiplex table entry. It includes the MultiplexTableEntryNumber and a list of MultiplexElements. A missing element list indicates that the entry is deactivated.

MultiplexElement: is a recursive structure that describes a single element and a repeat count. If of type logicalChannelNumber, the element indicates a single slot from the given logical channel and the repeat count indicates the length of the slot in octets. If of type subElementList, the element indicates a sequence of nested MultiplexElements and the repeat count indicates the number of times to repeat the sequence. In either case, if the repeatCount field is untilClosingFlag, this means to repeat the element indefinitely until the closing flag of the MUX-PDU.

In each MultiplexEntryDescriptor, the repeatCount of the final MultiplexElement in the elementList shall be set to "untilClosingFlag" and the repeatCount of all other MultiplexElements in the elementList shall be set to "finite". This ensures that all multiplex table entries define a multiplex sequence pattern of indefinite length, repeating until the closing flag of the MUX-PDU. A MultiplexEntryDescriptor with a missing elementList field shall indicate a deactivated entry.

Each MultiplexEntrySend request may contain up to 15 MultiplexEntryDescriptors, each describing a single multiplex table entry. Multiplex entries may be sent in any order.

B.4.2 Multiplex Entry Send Acknowledge

This is used to confirm receipt of one or more multiplexEntryDescriptors from a MultiplexEntrySend from the peer MTSE.

The sequenceNumber shall be the same as the sequenceNumber in the MultiplexEntrySend for which this is the confirmation.

multiplexTableEntryNumber indicates which multiplex table entries are being confirmed.

B.4.3 Multiplex Entry Send Reject

This is used to reject one or more multiplexEntryDescriptors from a MultiplexEntrySend from the peer MTSE.

The sequenceNumber shall be the same as the sequenceNumber in the MultiplexEntrySend for which this is the rejection.

MultiplexEntryRejectionDescriptions specifies which table entries are being rejected, and why. The causes of rejection are given in Table B.9.

Table B.9/H.245 – Reasons for rejecting a MultiplexEntrySend

ASN.1 codepoint	Cause
unspecified	No cause for rejection specified
descriptorTooComplex	The MultiplexEntryDescriptor exceeded the capability of the receive terminal.

B.4.4 Multiplex Entry Send Release

This is sent by the outgoing MTSE in the case of a timeout.

`multiplexTableEntryNumber` indicates which multiplex table entries have timed out.

B.5 Request Multiplex Table signalling messages

This set of messages is for the secure request of retransmission of one or more `MultiplexEntryDescriptors` from the transmitter to the receiver.

B.5.1 Request Multiplex Entry

This is used to request the retransmission of one or more `MultiplexEntryDescriptors`.

`entryNumbers` is a list of the `MultiplexTableEntryNumbers` of the `MultiplexEntryDescriptors` for which retransmission is requested.

B.5.2 Request Multiplex Entry Acknowledge

This is used by the incoming RMESE to indicate that the multiplex entry will be transmitted.

`entryNumbers` is a list of the `MultiplexTableEntryNumbers` of the `MultiplexEntryDescriptors` and will be transmitted.

B.5.3 Request Multiplex Entry Reject

This is used by the incoming RMESE to indicate that the multiplex entry will not be transmitted.

`entryNumbers` is a list of the `MultiplexTableEntryNumbers` of the `MultiplexEntryDescriptors` and will not be transmitted. The values of `MultiplexTableEntryNumber` in `entryNumbers` should match the values of `MultiplexTableEntryNumber` in `rejectionDescriptions` otherwise errors may occur during operation.

`RequestMultiplexEntryRejectionDescriptions` specifies which table entries are being rejected, and why. The causes of rejection are given in Table B.10.

Table B.10/H.245 – Reasons for rejecting a MultiplexEntrySend

ASN.1 codepoint	Cause
unspecified	No cause for rejection specified

B.5.4 Request Multiplex Entry Release

This is sent by the outgoing RMESE in the case of a timeout.

`entryNumbers` is a list of the `MultiplexTableEntryNumbers` of the `MultiplexEntryDescriptors` for which timeout has occurred.

B.6 Request Mode messages

This set of messages is used by a receive terminal to request particular modes of transmission from the transmit terminal.

B.6.1 Request Mode

This is used to request particular modes of transmission from the transmit terminal. It is a list, in order or preference (most preferable first), of modes that the terminal would like to receive. Each mode is described using a `ModeDescription`.

sequenceNumber is used to label instances of RequestMode so that the corresponding response can be identified.

ModeDescription: is a set of one or more ModeElements.

ModeElement: is used to describe a mode element, that is, one of the constituent parts of a complete mode description. It indicates the type of elementary stream that is requested and optionally how it is requested to be multiplexed.

type is used to indicate the type of elementary stream that is requested. It is a choice of VideoMode, AudioMode, DataMode, EncryptionMode, and H235Mode. H235Mode indicates that encrypted media is requested.

h223ModeParameters: is used to indicate parameters specific to using H.223 [10].

adaptationLayerType indicates which adaptation layer and options are requested for the requested type. The codepoints are as follows: nonStandard, al1Framed (AL1 framed mode), al1NotFramed (AL1 unframed mode), al2WithoutSequenceNumbers (AL2 with no sequence numbers present), al2WithSequenceNumbers (AL2 with sequence numbers present), and al3 (AL3, indicating the number of control field octets that will be present and the size of the send buffer, B_S , that will be used, the size being measured in octets), al1M (AL1M defined in Annex C/H.223 with specified parameters), al2M (AL2M defined in Annex C/H.223 with specified parameters), and al3M (AL3M defined in Annex C/H.223 with specified parameters).

segmentableFlag, when equal to true indicates that segmentable multiplexing is requested, and when equal to false indicates that non-segmentable multiplexing is requested.

h2250ModeParameters contains information specific for use along with H.225.0 and H.323.

redundancyEncodingMode (if present) specifies which *redundancyEncodingMethod* shall be used and which *secondaryEncoding* shall be used as redundancy encoding. The primary encoding is specified by the *type* element contained in the *ModeElement*.

B.6.1.1 Video Mode

This is a choice of VideoModes.

H261VideoMode: indicates the requested picture resolution (either QCIF or CIF), bit rate, in units of 100 bit/s, and still picture transmission.

H262VideoMode: indicates the requested profile and level, and the optional fields, if present, indicate the requested values of the parameters given. The optional fields are integers with units defined in Table B.2.

H263VideoMode: indicates the requested picture resolution (SQCIF, QCIF, CIF, 4CIF and 16CIF or some custom picture format) and bit rate, in units of 100 bit/s.

The booleans unrestrictedVector, arithmeticCoding, advancedPrediction, and pbFrames, when true, indicate that it is requested to use these optional modes that are defined in the annexes of Recommendation H.263.

The boolean errorCompensation, when true, indicates that the encoder is capable of processing videoNotDecodedMBs indications and compensating errors as illustrated in Appendix I/H.263. The encoder is not required to respond to videoNotDecoded indications. In a multipoint control unit (MCU), it may not be practical for the MCU to respond to all indications.

EnhancementOptions: indicates the requested scalability enhancement layer parameters.

H263Options: indicates the requested optional modes of H.263.

IS11172VideoMode: indicates request for constrainedBitstream and the optional fields, if present, indicate the requested values of the parameters given. The optional fields are integers with units defined in Table B.3.

B.6.1.2 Audio Mode

This is a choice of AudioModes.

The exact meaning of the G-series audio codepoints is given in Table B.4. There are four options for G.723.1 audio, to allow either of the bit rates (the low bit rate of 5.3 kbit/s or the high bit rate of 6.3 kbit/s) to be requested with or without the use of silence suppression.

G7231AnnexCMode: is used to request audio coded according to Annex C/G.723.1. maxAl-sduAudioFrames indicates the requested maximum number of audio frames per AL-SDU. The boolean silenceSupression, when true, requests the use of silence compression defined in Annex A/G.723.1. The fields of g723AnnexCAudioMode, highRateMode0, highRateMode1, lowRateMode0, lowRateMode1, sidMode0, and sidMode1 indicates the requested number of octets per frame for each of the audio and error protection modes of G.723.1 and Annex C/G.723.1.

IS11172AudioMode: is used to request audio coded according to ISO/IEC 11172-3 [44].

audioLayer indicates which coding layer is requested: either audioLayer1, audioLayer2 or audioLayer3.

audioSampling indicates which sample rate is requested: audioSampling32k, audioSampling44k1 and audioSampling48k indicate the audio sample rates 32 kHz, 44.1 kHz and 48 kHz respectively.

multichannelType indicates which multi-channel mode is requested: singleChannel, twoChannelStereo and twoChannelDual request single channel, stereo and dual channel operation respectively.

bitRate indicates the requested audio bit rate, and is measured in units of kbit/s.

IS13818AudioMode: is used to request audio coded according to ISO/IEC 13818-3 [45].

audioLayer indicates which coding layer is requested: either audioLayer1, audioLayer2 or audioLayer3.

audioSampling indicates which sample rate is requested: audioSampling16k, audioSampling22k05, audioSampling24k, audioSampling32k, audioSampling44k1 and audioSampling48k indicate the audio sample rates 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz and 48 kHz respectively.

multichannelType indicates which multi-channel mode is requested as specified in Table B.11.

Table B.11/H.245 – ISO/IEC 13818-3 multi-channel codepoints

ASN.1 codepoint	Semantic meaning of codepoint
singleChannel	One channel, using the 1/0 configuration. Single channel mode (as in ISO/IEC 11172-3).
twoChannelStereo	Two channels, using the 2/0 configuration, stereo channel mode (as in ISO/IEC 11172-3).
twoChannelDual	Two channels, using the 2/0 configuration, dual channel mode (as in ISO/IEC 11172-3).
threeChannels2-1	Three channels, using the 2/1 configuration. Left, Right and single surround channel.

Table B.11/H.245 – ISO/IEC 13818-3 multi-channel codepoints (*concluded*)

ASN.1 codepoint	Semantic meaning of codepoint
threeChannels3-0	Three channels, using the 3/0 configuration. Left, Centre and Right, without surround channel.
fourChannels2-0-2-0	Four channels, using the 2/0 + 2/0 configuration. Left and Right of the first programme and Left and Right of the second programme.
fourChannels2-2	Four channels, using the 2/2 configuration. Left, Right, Left surround and Right surround.
fourChannels3-1	Four channels, using the 3/1 configuration. Left, Centre, Right, and a single surround channel.
fiveChannels3-0-2-0	Five channels, using the 3/0 + 2/0 configuration. Left, Centre and Right of the first programme and Left and Right of the second programme.
fiveChannels3-2	Five channels, using the 3/2 configuration. Left, Centre, Right, Left surround and Right surround.

The boolean `lowFrequencyEnhancement`, when true, requests a low frequency enhancement channel.

The boolean `multilingual`, when true, requests up to seven multilingual channels.

`bitRate` indicates the requested audio bit rate, and is measured in units of kbit/s.

B.6.1.3 Data Mode

This is a choice of data applications and bit rates.

`bitRate` indicates the requested bit rate in units of 100 bit/s.

`t120` requests the use of the T.120 [32] protocol.

`dsm-cc` requests the use of the DSM-CC [46] protocol.

`userData` requests the use of unspecified user data from external data ports.

`t84` requests the use of T.84 [31] for the transfer of such images (JPEG, JBIG, Facsimile Gr.3/4).

`t434` requests the use of T.434 [34] for the transfer of telematic binary files.

`h224` requests the use of the real-time simplex device control protocol H.224 [11].

`nlpid` requests the use of the use of the specified network link layer data application.

`dsvdControl` requests the use of the DSVD terminal to support an out-of-band control channel.

`h222DataPartitioning` requests the use of the modified and restricted usage of data partitioning of H.262, as specified in Recommendation H.222.1, in which the enhancement data is transmitted as a data channel supported by the listed `DataProtocolCapability`.

`t30fax` requests the use of Annex C/T.30 analog mode (G3V), as specified in Recommendation T.39 for the DSVF/MSVF modes.

`t140` requests the use of the T.140 text conversation protocol, as specified in Recommendation T.140.

`t38fax` requests the use of Recommendation T.38 [29].

B.6.1.4 Encryption Mode

This is a choice of encryption modes.

h233Encryption requests the use of encryption according to Recommendations H.233 [14] and H.234 [15].

B.6.2 Request Mode Acknowledge

This is sent to confirm that the transmit terminal intends to transmit in one of the modes requested by the receive terminal.

The sequenceNumber shall be the same as the sequenceNumber in the RequestMode for which this is the confirmation.

The response field indicates the action from the remote terminal. The possible values of response are given in Table B.12.

Table B.12/H.245 – Confirmation responses to Request Mode

ASN.1 codepoint	Response
willTransmitMostPreferredMode	The transmit terminal will change to the receiver's most preferred mode.
willTransmitLessPreferredMode	The transmit terminal will change to one of the receiver's preferred mode, but not the most preferred mode.

B.6.3 Request Mode Reject

This is sent to reject the request by the receive terminal.

The sequenceNumber shall be the same as the sequenceNumber in the RequestMode for which this is the response.

The cause field indicates the reason for rejection of the requested mode. The cause values are given in Table B.13.

Table B.13/H.245 – Rejection responses to Request Mode

ASN.1 codepoint	Response
modeUnavailable	The transmit terminal will not change its mode of transmission as the requested modes are not available.
MultipointConstraint	The transmit terminal will not change its mode of transmission due to a multipoint constraint.
RequestDenied	The transmit terminal will not change its mode of transmission.

B.6.4 Request Mode Release

This is used by the outgoing MRSE in the case of a timeout.

B.7 Round Trip Delay messages

This set of messages is used by a terminal to determine the round trip delay between two communicating terminals. It also enables a H.245 user to determine whether the peer H.245 protocol entity is alive.

B.7.1 Round Trip Delay Request

This is sent from the outgoing RTDSE to the incoming RTDSE.

sequenceNumber is used to label instances of RoundTripDelayRequest so that the corresponding response can be identified.

B.7.2 Round Trip Delay Response

This is sent from the incoming RTDSE to the outgoing RTDSE.

The sequenceNumber shall be the same as the sequenceNumber in the RoundTripDelayRequest for which this is the response.

B.8 Maintenance Loop messages

This set of messages is used by a terminal to perform maintenance loop functions.

B.8.1 Maintenance Loop Request

This is sent to request a particular type of loop back. The types mediaLoop and logicalChannelLoop request the loopback of only one logical channel as indicated by LogicalChannelNumber, while the type systemLoop refers to all logical channels. The exact definition of these types is system specific and outside the scope of this Recommendation.

B.8.2 Maintenance Loop Acknowledge

This is used to confirm that the terminal will perform the loop as requested.

B.8.3 Maintenance Loop Reject

This is used to indicate that the terminal will not perform the loop as requested.

A terminal may use the cause canNotPerformLoop to indicate that it does not have the capability to perform the requested loop.

B.8.4 Maintenance Loop Command Off

On receipt of this command, the terminal shall disconnect all loops and restore audio, video and data paths to their normal condition.

B.9 Communication Mode Messages

This set of messages are used by an H.323 MC to convey the communication mode of an H.323 conference.

B.9.1 Communication Mode Command

The **CommunicationModeCommand** is sent by a H.323 MC to specify the communication mode for each media type: unicast or multicast. This command may cause a switch between a centralized and decentralized conference and therefore may involve closing all existing logical channels and opening new ones.

The **CommunicationModeCommand** specifies all the sessions in the conference. For each session, the following data is specified: the RTP session identifier, the associated RTP session ID if applicable, a terminal label if applicable, a description of the session, the datatype of the sessions (e.g. G.711), and a unicast or multicast address for the media and media control channels as appropriate for the conference configuration and type. In case a redundancy encoding shall be used, the communicationModeTableEntry also specifies the redundancyEncodingMethod as well as the secondary encoding format.

The **CommunicationModeCommand** conveys the transmit modes which conference endpoints are to use in a conference. The command does not convey receive modes, as they are specified by **OpenLogicalChannel** commands which are sent from the MC to the endpoints.

It is presumed that the **CommunicationModeCommand** is defining the modes of a conference and is therefore sent after the **multipointConference** indication which notifies an endpoint that it must comply with the commands of the MC. Endpoints should wait for a **CommunicationModeCommand** before opening logical channels when they have received a **multipointConference** indication.

Endpoints receiving a **CommunicationModeCommand** use the **terminalLabel** field of each table entry to determine if the entry is applicable for its own processing. Entries which do not contain a **terminalLabel** apply to all endpoints in the conference. Entries which contain **terminalLabel** are commands to specific endpoints and which match the **terminalLabel** in the entry. For example, when audio streams from all endpoints are placed on one multicast address (one session), the table entry for the audio mode, media address, and media control address will not contain a **terminalLabel**. When the table entry commands an endpoint to send its video to a multicast address, the MC will include that endpoint's **terminalLabel**.

SessionDependency is set by the MC to indicate when a session is dependent on another session for meaningful decoding of the data.

The **destination** field in the CommunicationModeTableEntry indicates the endpoint to which the transmitting endpoint should open a logical channel. If the **destination** field exists in the CommunicationModeTableEntry, the endpoint shall use the destination as the **destination** field in H2250LogicalChannelParameters of the OpenLogicalChannel message.

The CommunicationModeCommand can be used to instruct endpoints in a conference (or a point-to-point call) to change modes (by indicating a new mode with the mediaChannel already used) or to transmit to a new address (by indicating the mode currently in use, but with new mediaChannel). Similarly, an endpoint that receives a CommunicationModeCommand indicating the mode currently in use and no mediaChannel should close the appropriate channel and the attempt to reopen using the OpenLogicalChannel procedures, where the OpenLogicalChannelAck contains the address to which the endpoint will send the medium.

B.9.2 Communication Mode Request

This is sent to the MC to request the communication mode of the current conference.

B.9.3 Communication Mode Response

This is sent by the MC, in response to a CommunicationModeRequest to specify the communication mode of a conference.

B.10 Conference Request and Response Messages

TerminalID, which is used in the Conference Request and Response Messages, has a length of 128 octets. When communicating between an H.323 terminal and an H.320 terminal via a H.323 Gateway, this field will be truncated to 32 octets.

B.10.1 Terminal List Request

This request equates to H.230 TCU as described in Recommendation H.243.

B.10.2 Terminal List Response

This request equates to a sequence of terminalNumbers as described in Recommendation H.230.

B.10.3 Make Me Chair

This request equates to CCA as described in Recommendation H.230.

B.10.4 Cancel Make Me Chair

This request equates to CIS as described in Recommendation H.230.

B.10.5 Make Me Chair Response

This request equates to either H.230 CIT if the chair control token is granted or H.230 CCR if the chair control token is denied.

B.10.6 Drop Terminal

This request equates to CCD as described in Recommendation H.230.

B.10.7 Terminal Drop Reject

This response equates to CIR as described in Recommendation H.230.

B.10.8 Request Terminal ID

This request equates to TCP as described in Recommendation H.230.

B.10.9 MC Terminal ID Response

This response equates to TIP as described in Recommendation H.230.

B.10.10 Enter H.243 Password Request

This request equates to TCS1 as described in Recommendation H.230.

B.10.11 Password Response

This response equates to IIS as described in Recommendation H.230.

B.10.12 Enter H.243 Terminal ID Request

This request equates to TCS2/TCI as described in Recommendation H.230.

B.10.13 Terminal ID Response

This response equates to IIS as described in Recommendation H.230.

B.10.14 Enter H.243 Conference ID Request

This request equates to TCS3 as described in Recommendation H.230.

B.10.15 Conference ID Response

This response equates to IIS as described in Recommendation H.230.

B.10.16 Video Command Reject

This request equates to VCR as described in Recommendation H.230.

B.10.17 Enter Extension Address Request

This request equates to TCS4 as described in Recommendation H.230.

B.10.18 Extension Address Response

This response equates to IIS as described in Recommendation H.230.

B.10.19 Request Chair Control Token Owner

This request equates to TCA as described in Recommendation H.230 for the Chair Control Token.

B.10.20 Chair Token Owner Response

This response equates to TIR as described in Recommendation H.230 for the Chair Control Token.

B.10.21 Request Terminal Certificate

This request is issued by any endpoint in a conference to its MC. It allows an endpoint to obtain the digital certificate for the user at a particular terminal. The requesting terminal may optionally include its own terminalCertificate and a challengeString which is encrypted with the private key.

The CertSelectionCriteria defines a set certificates that are acceptable to the requester. The responder (the MC) should attempt to satisfy these criteria. CertSelectionCriteria may be present along with terminalLabel. In this case, the MC may use the criteria either to select an appropriate certificate from those presented by the specified terminal, or may request from the specified terminal for a certificate matching the criteria, which it then returns to the original, requesting terminal.

This response may return the digital certificate and optionally a signature associated with the certificate as per the following:

- If the source of the terminalCertificateResponse does not have a suitable certificate, this message may be returned with no certificate (and therefore, no certificateResponse structure).
- If an endpoint is requesting the certificate of another endpoint in a multipoint conference (indicated by the terminalLabel), the responding MC shall return a certificate associated with the requested endpoint (contained within the certificateResponse structure).
- The certificateResponse structure should be present. In the event that the MC is presenting the certificate to the requester on behalf of another endpoint, there shall be a cryptographic link between the signatures and that of the MC. This shall be provided in one of two ways:
 - The private key used to protect session key material distributed in the most recent exchange shall be used.
 - If there has been no key material exchanged, or that key is not suitable for signing, the certificate that was used during the most recent endpoint-MC authentication shall be the source of the private key.

B.10.22 Terminal Certificate Response

This response returns the digital certificate and a responseString which is encrypted with the private key for a specific terminal.

B.10.23 Broadcast My Logical Channel

This request is similar to H.230 MCV but only refers to a single logical channel and has a response message broadcastMyLogicalChannelResponse which acknowledges the request.

B.10.24 Broadcast My Logical Channel Response

Provides a granted or denied response to the BroadcastMyLogicalChannel Request.

B.10.25 Make Terminal Broadcaster

This request is similar to H.230 VCB and has a response message makeTerminalBroadcasterResponse which acknowledges the request.

B.10.26 Make Terminal Broadcaster Response

Provides a granted or denied response to the MakeTerminalBroadcaster Request.

B.10.27 Send This Source

This request is similar to H.230 VCS and has a response message SendThisSourceResponse which acknowledges the request.

B.10.28 SendThis Source Response

Provides a granted or denied response to SendThisSource Request.

B.10.29 Request All Terminals Ids

Sent by an endpoint to the MC of a conference to obtain all of the terminal labels and terminal IDs of the conference participants.

B.10.30 Request All Terminal Ids Response

Response to RequestAllTerminalIDs containing a list of all endpoints in conference by terminalLabel and terminalID.

B.10.31 RemoteMC Request

This request is sent by an active MC to another MC to activate/deactivate it. A RemoteMC Request with a choice of masterActivate or slaveActivate may be sent by an active MC to an inactive MC to activate it as Master or Slave, respectively, of a cascaded connection. A RemoteMC Request with a choice of deActivate may be sent by a Master MC to an already active Slave MC to deactivate it.

B.10.32 RemoteMC Response

The RemoteMC Response is sent to indicate acceptance or rejection of the RemoteMC Request. Acceptance of this request is determined by the following criteria:

Choice = activateSlave:

The receiver is inactive and the sender of the request initiated this call with a conferenceGoal of INVITE in the H.225 Setup message or the receiver of the request initiated this call with a conferenceGoal of JOIN in the H.225 Setup message.

Choice = activateMaster:

The receiver is inactive and the sender of the request initiated this call with a conferenceGoal of CREATE in the H.225 Setup message.

Choice = deActivate:

The receiver is an active MC.

If the conditions stated above are not met, the request should be rejected with a choice of invalidConfiguration.

A reject choice of functionNotSupported is used by endpoints not supporting cascading.

B.11 Multilink Messages

multilinkRequest, multilinkResponse, and multilinkIndication messages are used to support the use of channel aggregation according to Recommendation H.MULTILINK, as specified in Annex F/H.324. These messages provide for the addition and removal of physical connections,

automatic exchange of network addresses (telephone numbers), and control of H.MULTILINK operation.

B.11.1 callInformation Request and Response

MultilinkRequest.callInformation is used by the initiator, as defined in Annex F/H.324, to request the information needed to establish and associate additional physical connections. The maximum number of additional connections that the sender is capable of establishing is sent in the maxNumberOfAdditionalConnections parameter.

The MultilinkResponse.callInformation message includes the DiallingInformation parameter, with contents as described below, as well as a callAssociationNumber. The callAssociationNumber shall contain a 32-bit uniformly distributed random number. Any subsequent callInformation exchanges within the same session shall use the identical callAssociationNumber.

B.11.2 addConnection Request and Response

The MultilinkRequest.addConnection message may be used by the responder, as defined in Annex F/H.324, to ask the initiator to add physical connections. The DiallingInformation structure shall indicate the connections to be added. The sequenceNumber parameter shall be incremented by 1, modulo 256, for each new MultilinkRequest.addConnection message sent.

On receiving this message, the initiator shall respond with a MultilinkResponse.addConnection message indicating that it either intends to add the connections as requested, or that it does not intend to do so, along with the appropriate reason code. The sequenceNumber parameter shall be equal to the sequenceNumber parameter of the corresponding MultilinkRequest.addConnection message.

B.11.3 removeConnection Request and Response

The MultilinkRequest.removeConnection message may be used by either the initiator or responder, as defined in Annex F/H.324, to request that the far-end remove a channel from the H.MULTILINK channel set. This is used as part of the procedure in Annex F/H.324 for removing physical connections. The connectionIdentifier parameter shall identify the channel to be removed, using the channel numbering received via H.MULTILINK from the terminal receiving the MultilinkRequest.removeConnection message.

The MultilinkResponse.removeConnection message shall be sent in response, after the channel has been removed from the H.MULTILINK channel set, indicating that this channel is no longer (or was never) in use. The connectionIdentifier parameter shall be identical to the value in the corresponding MultilinkRequest.removeConnection message.

B.11.4 maximumHeaderInterval Request and Response

The MultilinkRequest.maximumHeaderInterval message may be used to request the actual H.MULTILINK Maximum Header Interval being used by the remote transmitter without altering it (the currentIntervalInformation choice), or to request a particular value to be used instead (the requestedInterval choice, with units in milliseconds).

The MultilinkResponse.maximumHeaderInterval message shall be sent in response. If the corresponding request was a request for information about the current minimum rate, the terminal shall provide the value that its transmitter is currently using as the Maximum Header Interval in the response. If the corresponding request specified a particular minimum rate to use, the terminal should attempt to comply with this request by modifying the Maximum Header Interval used by its transmitter. Whether or not it makes a change to the Maximum Header Interval, the response shall indicate the new value that is in use (which may be different from the requested value).

B.11.5 Multilink Indications

The MultilinkIndication.crcDesired message may be sent by a terminal to indicate its desire that the remote terminal send the optional H.MULTILINK data CRC in all subsequent data sets. The receiving terminal may optionally comply: no explicit acknowledgment or response is required.

The MultilinkIndication.excessiveError message may be sent to indicate to the remote terminal that excessive errors are being received on a particular connection. The means for the terminal to determine the error rate or the criterion for determining what is excessive is defined locally at that terminal. The connection is indicated using the connectionIdentifier parameter. On receipt of this message, a terminal may choose to take corrective action. The particular corrective action that it should take is not specified.

B.11.6 DiallingInformation

The DiallingInformation type is used to provide explicit dialling information (telephone numbers) to allow the automatic establishment of physical connections. The differential choice provides a list of DiallingInformationNumber parameters, one for each potential additional connection. The length of this list indicates the maximum number of additional connections available. If such information is not available, the infoNotAvailable choice is used, indicating only the number of additional connections that are available.

B.11.7 DiallingInformationNumber

The DiallingInformationNumber type includes up to three subparameters that indicate the dialling information for a physical connection differentially relative to the corresponding information for an already established initial connection.

The networkAddress parameter shall include the least significant (right-most) portion of the telephone number for the connection, up to and including the most significant digit that is different from the number for an initially established connection, and shall include no digits that are more significant than this. If the number for the connection is identical to that of the initial connection, the networkAddress parameter shall consist of a zero-length string (since there are no differing digits in the telephone number).

NOTE – The differential digit method is used instead of the full E.164 digit string because the first few digits of the number to be dialled can vary based on the geographic location of the two terminals; for example whether or not they are located in the same city.

If there is a subaddress used for dialling, and the subaddress of a given connection is different from that of the initial connection, the responder shall include the subaddress, in full, in the optional subAddress parameter.

The network types supported for the connection (GSTN, ISDN, or both) shall be indicated using the networkType parameter.

B.11.8 DiallingInformationNetworkType

The DiallingInformationNetworkType type indicates a circuit-switched network type, n-isdn (N-ISDN) or gsn (GSTN).

B.11.9 ConnectionIdentifier

The ConnectionIdentifier type is used to identify uniquely a single physical connection in H.MULTILINK, using a combination of channelTag and sequenceNumber from an H.MULTILINK Header. If a Channel Tag was not specified at all in the Header, a value of zero shall be used for the channelTag parameter.

B.11.10 New ATM Virtual Channel Command

This is used to command the remote terminal to open an ATM virtual channel with the given parameters.

resourceID is used to identify the ATM virtual channel. The means by which this parameter is associated with an ATM virtual channel is not specified in this Recommendation.

bitRate indicates the bit rate, measured at the AAL-SAP, of the virtual channel, and is measured in units of 64 kbit/s.

bitRateLockedToPCRClock indicates that the bit rate of the virtual channel is clocked to the clock used to produce H.222.0 clock reference values (Program clock reference or System clock reference).

bitRateLockedToNetworkClock indicates that the bit rate of the virtual channel is clocked to the local network clock. This does not guarantee that the bit rate clock will be locked to the local network at the receiver, as common network clocks may not be available.

aal indicates which ATM Adaptation Layer will be used, and its parameters.

The sequence aal1 indicates which of the options for ATM adaptation layer 1, as specified in Recommendation I.363 [25], are supported. The codepoints are defined in Table B.1.

The sequence aal5 indicates which of the options for ATM adaptation layer 5, as specified in Recommendation I.363 [25], are supported. forwardMaximumSDUSize and backwardMaximumSDUSize indicate the maximum CPCS-SDU size in the forward and reverse directions, measured in octets.

multiplex indicates the type of multiplex that will be used on the ATM virtual channel. The options are noMultiplex (no H.222.0 multiplex), H.222.0 Transport Stream and H.222.0 Program Stream.

B.12 Commands

A command message requires action but no explicit response.

B.12.1 Send Terminal Capability Set

specificRequest commands the far end terminal to indicate its transmit and receive capabilities by sending one or more TerminalCapabilitySets that contain the information requested, as specified below. This command may be sent at any time to elicit the capabilities of the remote terminal, for example, following an interruption or other cause for uncertainty; however, such messages should not be sent repetitively without strong cause.

A terminal shall only request the transmission of capabilityTableEntryNumbers and capabilityDescriptorNumbers that it has previously received. A terminal shall ignore any requests to transmit capabilityTableEntryNumbers and capabilityDescriptorNumbers that it has not previously transmitted and no fault shall be considered to have occurred.

The boolean multiplexCapability, when true, requests the transmission of the MultiplexCapability.

capabilityTableEntryNumbers is a set of the CapabilityTableEntryNumbers that indicate the CapabilityTableEntries that the terminal requests to be transmitted.

capabilityDescriptorNumbers is a set of the capabilityDescriptorNumbers that indicate the CapabilityDescriptors that the terminal requests to be transmitted.

genericRequest commands the far end terminal to send its entire terminal capability set.

B.12.2 Encryption

This command is used to exchange encryption capabilities and to command the transmission of an Initialization Vector (IV); refer to Recommendations H.233 [14] and H.234 [15].

encryptionSE is an H.233 Session Exchange (SE) message, except that the error protection bits described in Recommendation H.233 shall not be applied.

encryptionIVRequest commands the far-end encryptor to transmit a new IV in a logical channel opened for encryptionData.

encryptionAlgorithmID indicates to the receiver that the sending terminal will associate the given h233AlgorithmIdentifier value with the non-standard encryption algorithm associatedAlgorithm.

B.12.3 Flow Control

This command is used to specify the upper limit of bit rate of either a single logical channel or the whole multiplex. A terminal may send this command to restrict the bit rate that the far-end terminal sends. A terminal that receives this command shall comply with it.

When scope is of type logicalChannelNumber the limit applies to the given logical channel, when scope is of type resourceID the limit applies to the given ATM virtual channel, and when scope is of type wholeMultiplex the limit applies to the whole multiplex.

maximumBitRate is measured in units of 100 bit/s averaged over non-overlapping consecutive periods of one second. When this is present, the specified limit supersedes any previous limit, whether higher or lower. When it is not present any previous restriction on the bit rate for the channel is no longer applicable.

The point at which the bit rate limit is applied, and the specification of which bits are included in the calculation of bit rate is not specified in this Recommendation, but should be specified by Recommendations that use this Recommendation.

Each transmission of this command affects a specific logical channel or the entire multiplex. More than one such command may be in effect at the same time, up to the number of open logical channels plus one, for the overall multiplex limitation.

NOTE – When the bit rate that can be transmitted on a logical channel is constrained to particular values, for example G.723.1 audio, and the request is to transmit at a rate lower than the lowest rate at which it would normally operate, it shall respond by stopping transmission on the logical channel.

B.12.4 End session

This command indicates the end of the H.245 session. After transmitting EndSessionCommand, the terminal shall not send any more of the messages defined in this Recommendation.

disconnect indicates that the connection will be dropped.

gstnOptions: is a choice of alternatives that will occur after ending the H.245 session when a V-series modem is used over the GSTN.

The possible options are given in Table B.14.

**Table B.14/H.245 – Options after EndSessionCommand
when using a V-series modem over the GSTN**

ASN.1 codepoint	Option
telephonyMode	The terminal shall initiate the clear-down procedures defined in the V-series modem Recommendation, except that it shall not physically disconnect the GSTN connection.
v8bis	The terminal shall initiate the clear-down procedures defined in the V-series modem Recommendation and enter a V.8 <i>bis</i> session.
v34DSVD	The terminal shall preserve the V.34 modem connection, but use it to support V.DSVD.
v34DuplexFAX	The terminal shall preserve the V.34 modem connection, but use it to support T.30 FAX [27].
v34H324	The terminal shall preserve the V.34 modem connection, but use it to support H.324 [24].

isdnOptions: is a choice of alternatives that will occur after ending the H.245 session when a digital communications terminal is used over a digital network.

The possible options are given in Table B.15.

**Table B.15/H.245 – Options after EndSessionCommand when using
a digital communications terminal over a digital network**

ASN.1 codepoint	Option
telephonyMode	The terminal shall initiate the clear-down procedures defined in the Recommendation governing communication on the particular digital channel to which the terminal is connected, except that it shall not physically disconnect the digital connection.
v140	The terminal shall initiate the clear-down procedures defined in the Recommendation governing communication on the particular digital channel to which the terminal is connected and enter a V.140 session.
TerminalOnHold	The terminal shall initiate the "terminal on hold" procedures defined in the Recommendation governing communication on the particular digital channel to which the terminal is connected.

B.12.5 Miscellaneous Command

This is used for a variety of commands, some of which are present in Recommendations H.221 [7] and H.230 [13].

logicalChannelNumber indicates the logical channel number to which the command applies. It shall indicate a logical channel opened for video data when the type is one of videoFreezePicture, videoFastUpdatePicture, videoFastUpdateGOB, videoTemporalSpatialTradeOff, videoSendSyncEveryGOB, videoFastUpdateMB, and videoSendSyncEveryGOBCancel. When the type is one of equaliseDelay, zeroDelay, multipointModeCommand or cancelMultipointModeCommand where multiple logical channels are involved, the logicalChannelNumber shall be arbitrary, but shall be a valid LogicalChannelNumber (i.e. in the range 1-65535) and the receiver shall ignore the value.

equaliseDelay and zeroDelay shall have the same meaning as the commands ACE and ACZ defined in Recommendation H.230 [13].

multipointModeCommand commands that a terminal in receipt shall comply with all requestMode requests issued by the MCU. An example of a mode change is an audio coding change from G.711 to G.728.

cancelMultipointModeCommand cancels a previously sent multipointModeCommand command.

videoFreezePicture commands the video decoder to complete updating the current video frame and subsequently display the frozen picture until receipt of the appropriate freeze-picture release control signal.

videoFastUpdatePicture commands the video encoder to enter the fast-update mode at its earliest opportunity.

videoFastUpdateGOB commands the far-end video encoder to perform a fast update of one or more GOBs. firstGOB indicates the number of the first GOB to be updated, and numberOfGOBs indicates the number of GOBs to be updated. It shall only be used with video compression algorithms that define GOBs, for example, Recommendations H.261 and H.263.

videoTemporalSpatialTradeOff commands the far-end video encoder to change its trade-off between temporal and spatial resolution. A value of 0 commands a high spatial resolution and a value of 31 commands a high frame rate. The values from 0 to 31 indicate monotonically a desire for higher frame rate. Actual values do not correspond to precise values of spatial resolution or frame rate.

videoSendSyncEveryGOB commands the far-end video encoder to use sync for every GOB as defined in Recommendation H.263 [20], until the command videoSendSyncEveryGOBCancel is received, from which time the far-end video encoder may decide the frequency of GOB syncs. These commands shall only be used with video encoded according to Recommendation H.263.

videoFastUpdateMB commands the far-end video encoder to perform a fast update of one or more MBs. firstGOB indicates the number of the first GOB to be updated and is only relative to H.263, firstMB indicates the number of the first MB to be updated and is only relative to H.261 and numberOfMBs indicates the number of MBs to be updated. It shall only be used with video compression algorithms that define MBs, for example, H.261 and H.263. Terminals may respond to this command with a GOB update which includes the MBs requested.

maxH223MUXPDUsizes commands the transmitter to restrict the size of the H223 MUX-PDUs that it transmits to a maximum of the specified number of octets.

encryptionUpdate and EncryptionUpdateRequest are used to initiate and distribute new keying material to be used in the encryption of the indicated media channels.

Switch receive media on and off can be used by an MC to command an endpoint to switch between a unicast and multicast channel when the MC+MP is mixing audio. In this case, when the MC stream includes the terminal audio, the MC+MP can switch the endpoint to a unicast stream which would contain a special mix for the terminal with its audio removed.

switchReceiveMediaOff is used by an MC to indicate to an endpoint that a particular logical channel should not be used for receive media.

switchReceiveMediaOn is used by an MC to indicate to an endpoint that a particular logical channel should be used for receive media.

doOneProgression commands the video encoder to begin producing a progressive refinement sequence. In this mode, the encoder produces video data consisting of one picture followed by a sequence of zero or more frames of refinement of the quality of the same picture. The encoder stays in this mode until either the encoder decides an acceptable fidelity level has been reached or the progressiveRefinementAbortOne command is received. In addition, the encoder shall insert the Progressive Refinement Segment Start Tag and the Progressive Refinement Segment End Tag to

mark the beginning and end of the progressive refinement as defined in the Supplemental Enhancement Information Specification of Annex L/H.263.

doContinuousProgressions commands the video encoder to begin producing progressive refinement sequences. In this mode, the encoder produces video data consisting of one picture followed by a sequence of zero or more frames of refinement of the quality of the same picture. When the encoder decides an acceptable fidelity level has been reached or the progressiveRefinementAbortOne command is received, the encoder stops refining the current progression and begins another progressive refinement for a different picture. The sequence of progressive refinements continues until the progressiveRefinementAbortContinuous command is received. In addition, the encoder shall insert Progressive Refinement Segment Start Tags and Progressive Refinement Segment End Tags to mark the start and end of each progressive refinement as defined in the Supplemental Enhancement Information Specification of Annex L/H.263.

doOneIndependentProgression commands the video encoder to begin an independent progressive refinement sequence. In this mode, the encoder produces video data consisting of one Intra picture followed by a sequence of zero or more frames of refinement of the quality of the same picture. The encoder stays in this mode until either the encoder decides an acceptable fidelity level has been reached or the progressiveRefinementAbortOne command is received. In addition, the encoder shall insert the Progressive Refinement Segment Start Tag and the Progressive Refinement Segment End Tag to mark the beginning and end of the progressive refinement as defined in the Supplemental Enhancement Information Specification of Annex L/H.263.

doContinuousIndependentProgressions commands the video encoder to begin producing independent progressive refinement sequences. In this mode, the encoder produces video data consisting of one Intra picture followed by a sequence of zero or more frames of refinement of the quality of the same picture. When the encoder decides an acceptable fidelity level has been reached or the progressiveRefinementAbortOne command is received, the encoder stops refining the current progression and begins another independent progressive refinement for a different picture. The sequence of independent progressive refinements continues until the progressiveRefinementAbortContinuous command is received. In addition, the terminal shall insert Progressive Refinement Segment Start Tags and Progressive Refinement Segment End Tags to mark the start and end of each independent progressive refinement as defined in the Supplemental Enhancement Information Specification of Annex L/H.263.

progressiveRefinementAbortOne commands the video encoder to terminate doOneProgression, doOneIndependentProgression, or the current progressive refinement in the sequence of progressive refinements in either doContinuousProgressions or doContinuousIndependentProgressions.

progressiveRefinementAbortContinuous commands the video encoder to terminate either doContinuousProgressions or doContinuousIndependentProgressions.

B.12.6 Conference Command

BroadcastMyLogicalChannel shall be similar to H.230 MCV but shall only refer to a single logical channel.

CancelBroadcastMyLogicalChannel shall be similar to H.230 Cancel-MCV but shall only refer to a single logical channel.

MakeTerminalBroadcaster shall be defined as H.230 VCB.

CancelMakeTerminalBroadcaster shall be defined as Cancel-VCB.

SendThisSource shall be defined as H.230 VCS.

CancelSendThisSource shall be defined as H.230 Cancel-VCS.

DropConference shall be defined as H.230 CCK.

Substitute CID Command allows an active MC to change the Conference Identifier (CID), effectively moving the recipient of this command into another conference. The recipient of this command shall use the newly assigned CID in all future call signalling messages.

B.12.7 H.223 Multiplex Reconfiguration

h223ModeChange commands the transmitter to change the level of multiplex mode as described in Annex C/H.324, to level 0, level 1, level 2, or level 2 with Annex B/H.223 optional header.

h223AnnexADoubleFlag commands the transmitter to start or stop the use of double-flag mode of Annex A/H.223.

B.13 Indications

An indication contains information that does not require action or response.

B.13.1 Function Not Understood

This is used to return requests, responses and commands that are not understood to the transmitter.

If a terminal receives a request, response or command that it does not understand, either because it is non-standard or has been defined in a subsequent revision of this Recommendation, it should respond by sending FunctionNotSupported or FunctionNotUnderstood.

NOTE – FunctionNotUnderstood was named FunctionNotSupported in version 1 of Recommendation H.245. The name of this function was changed to allow for the addition of a more powerful FunctionNotSupported command without breaking backward compatibility with version 1 syntax.

B.13.2 Miscellaneous Indication

This is used for a variety of indications, some of which are present in Recommendations H.221 [7] and H.230 [13].

logicalChannelNumber indicates the logical channel number to which the indication applies. It shall indicate a logical channel opened for video data when the type is videoIndicateReadyToActivate, and videoTemporalSpatialTradeOff. When the type is one of multipointConference, cancelMultipointConference, multipointZeroComm, cancelMultipointZeroComm, multipointSecondaryStatus, or cancelMultipointSecondaryStatus where multiple logical channels are involved, the logicalChannelNumber shall be arbitrary, but shall be a valid LogicalChannelNumber (i.e. in the range 1-65535) and the receiver shall ignore the value.

logicalChannelInactive is used to indicate that the content of the logical channel does not represent a normal signal. It is analogous to AIM and VIS defined in Recommendation H.230.

logicalChannelActive is complementary to logicalChannelInactive. It is analogous to AIA and VIA defined in Recommendation H.230. MultipointZeroComm, cancelMultipointZeroComm, multipointSecondaryStatus, and cancelMultipointSecondaryStatus shall have the same meaning as MIZ, cancelMIZ, MIS and cancelMIS respectively, as defined in Recommendation H.230.

multipointConference indicates that the terminal is joined to an H.243 multipoint conference, and the terminal is expected to obey bit rate symmeterization. However, bit rate symmeterization will be enforced via FlowControlCommand messages. Note that multipointConference has exactly the same meaning as MCC in Recommendation H.230. Note that multipointConference, like MCC, does not require mode symmetry.

videoIndicateReadyToActivate shall have the same meaning as VIR defined in Recommendation H.230, that is, it is transmitted by a terminal whose user has decided not to send video unless he will also receive video from the other end.

videoTemporalSpatialTradeOff indicates to the far-end video decoder its current trade-off between temporal and spatial resolution. A value of 0 indicates a high spatial resolution and a value of 31 indicates a high frame rate. The values from 0 to 31 indicate monotonically a higher frame rate. Actual values do not correspond to precise values of spatial resolution or frame rate. A terminal that has indicated temporalSpatialTradeOffCapability shall transmit this indication whenever it changes its trade-off and when a video logical channel is initially opened.

videoNotDecodedMBs indicates to the far-end video encoder that a set of MBs has been received erroneously and that any MB in the specified set has been treated as not coded. The encoder may use this information to compensate transmission errors, as illustrated in Appendix I/H.263. firstMB indicates the number of the first MB treated as not coded, and numberOfMBs indicates the number of MBs treated as not coded. The MB numbering is done according to H.263. The temporal reference of the picture containing not decoded MBs is indicated in temporalReference. This indication shall only be used with the H.263 video compression algorithm.

B.13.3 Jitter Indication

This is used to indicate the amount of jitter, as estimated by the receive terminal, of a logical channel. It may be useful for choice of bit-rate and buffer control in video channels, or to determine an appropriate rate of transmission of timing information, etc. The video encoder will then have the option of using this information to restrict the video bit-rate or the video decoder buffer fluctuations to help prevent decoder buffer underflow or overflow, given the occurring jitter. If the encoder takes this option, it will enable correct operation for existing designs of video decoder buffers, regardless of the amplitude of received jitter, as well as allow correct operation with minimum delay.

When scope is of type logicalChannelNumber, the information applies to the given logical channel, when scope is of type resourceID, the information applies to the given ATM virtual channel and when scope is of type wholeMultiplex, the information applies to the whole multiplex.

estimatedReceivedJitterMantissa and estimatedReceivedJitterExponent provide an estimate of the jitter that has been received by the terminal that has sent the message.

estimatedReceivedJitterMantissa indicates the mantissa of the jitter estimate as given in Table B.16.

Table B.16/H.245 – Mantissa of estimatedReceivedJitterMantissa in JitterIndication

estimatedReceivedJitterMantissa	Mantissa
0	1
1	2.5
2	5
3	7.5

estimatedReceivedJitterExponent indicates the exponent of the jitter estimate as given in Table B.17.

Table B.17/H.245 – Exponent of estimatedReceivedJitterExponent in JitterIndication

estimatedReceivedJitterExponent	Exponent
0	Out of range
1	1 μ s
2	10 μ s
3	100 μ s
4	1 ms
5	10 ms
6	100 ms
7	1 s

The jitter estimate is obtained by multiplying the mantissa by the exponent, unless estimatedReceivedJitterExponent is equal to zero, in which case the estimate is just known to be more than 7.5 seconds.

skippedFrameCount indicates how many frames have been skipped by the decoder since the last JitterIndication message was received. Since the maximum value that can be encoded is 15, if this option is implemented, this information must be transmitted before more than 15 frames have been skipped.

NOTE – Since frames are skipped when the decoder buffer underflows, additional jitter may cause the decoder buffer to underflow more or less often than the encoder expects frame skips to happen.

additionalDecoderBuffer indicates the additional size of the video decoder buffer over and above that required by the indicated profile and level. This is defined in the same way as vbv_buffer_size H.262 [19].

B.13.4 H.223 Skew Indication

This is used to indicate to the far-end terminal the average amount of time skew between two logical channels.

logicalChannelNumber1 and logicalChannelNumber2 are logical channel numbers of opened logical channels.

skew is measured in milliseconds, and indicates the delay that must be applied to data belonging to logicalChannelNumber2 as measured at the output of the multiplex, to achieve synchronization with logicalChannelNumber1 as measured at the output of the multiplex. The skew includes differences in: sample time, encoder delay and transmitter buffer delay, and is measured relative to the transmission time of the first bit of data representing a given sample point. The actual delay necessary for synchronization is dependent on decoder implementation, and is a local matter for the receiver.

B.13.5 New ATM Virtual Channel Indication

This is used to indicate the parameters of an ATM virtual channel that the terminal intends to open.

resourceID is used to identify the ATM virtual channel. The means by which this parameter is associated with an ATM virtual channel is not specified in this Recommendation.

bitRate indicates the bit rate, measured at the AAL-SAP, of the virtual channel, and is measured in units of 64 kbit/s.

`bitRateLockedToPCRClock` indicates that the bit rate of the virtual channel is clocked to the clock used to produce H.222.0 clock reference values (Program clock reference or System clock reference).

`bitRateLockedToNetworkClock` indicates that the bit rate of the virtual channel is clocked to the local network clock. This does not guarantee that the bit rate clock will be locked to the local network at the receiver, as common network clocks may not be available.

`aal` indicates which ATM Adaptation Layer will be used, and its parameters.

The sequence `aal1` indicates which of the options for ATM adaptation layer 1, as specified in Recommendation I.363 [25], are supported. The codepoints are defined in Table B.1.

The sequence `aal5` indicates which of the options for ATM adaptation layer 5, as specified in Recommendation I.363 [25], are supported. `forwardMaximumSDUSize` and `backwardMaximumSDUSize` indicate the maximum CPCS-SDU size in the forward and reverse directions, measured in octets.

`multiplex` indicates the type of multiplex that will be used on the ATM virtual channel. The options are `noMultiplex` (no H.222.0 multiplex), H.222.0 Transport Stream and H.222.0 Program Stream.

B.13.6 User Input

This is used for User Input messages.

`alphanumeric` is a string of characters coded according to T.51 [30]. This could be used for keypad input, an equivalent to DTMF.

userInputSupportIndication: indicates to the remote terminal which GENERALSTRING types the terminal supports.

NOTE 1 – It is expected that most implementations of PER decoders will not be capable of decoding other strings than IA5. This indication should be used to "warn" the remote terminal not to attempt fancy variable length coding schemes.

`nonStandard` is a NonStandardParameter indicating a non-standard use of the UserInput indication message.

The boolean `basicString`, when true, indicates that the characters 0-9, * and # are supported.

The boolean `ia5String`, when true, indicates that the complete IA5String character set is supported.

The boolean `generalString`, when true, indicates that the complete GeneralString character set is supported.

NOTE 2 – Any data that is carried in this Recommendation, including user input messages, may not be encrypted.

The **signal** and **signalUpdate** indications may be used when precise control is desired over the alignment of DTMF or hookflash with audio in the associated logical channel and when control or indication of the duration of DTMF is needed.

signal indicates the signalling element to be produced when sent to a PSTN gateway, that was detected in the audio stream when sent from a PSTN gateway, or to be signalled between other endpoint combinations. When received by a gateway to the PSTN, **signal** causes the gateway to inject the specified signalling element into the PSTN channel; when received by a gateway to another H-series terminal, **signal** will be translated to the appropriate message in the protocol of the connected terminal. Gateways produce **signal** (and **signalUpdate**) messages to indicate detection of signalling elements in the audio received from a PSTN endpoint, or by translation of corresponding messages from other protocols.

signalType is set to "!" (exclamation point) to indicate a hookflash, or to one of "0123456789*#ABCD" to indicate a DTMF tone.

NOTE 3 – Hookflash is a momentary on-hook condition (typically one-half second in duration), commonly used to control features in the attached switching equipment. It may not be possible for a gateway to produce or detect a hookflash due to characteristics of the PSTN channel or due to local configuration (to prevent undesired activation of features in attached equipment). Therefore, the ability to transmit or receive hookflash indications is separately declared in **UserInputCapability**.

duration indicates the total duration of the tone if known, or an initial estimate of the tone duration if the tone continues to be in progress at the time **signal** is transmitted. If **duration** is omitted, the receiver shall use an appropriate default based on local configuration and network requirements. **duration** shall be ignored in the case of a hookflash ("!") indication.

signalUpdate revises the estimate of the total duration or declares the actual measured duration of the tone detected or to be generated. It should be transmitted so as to arrive well before the estimate that was previously sent in **signal** or **signalUpdate** expires; otherwise, the revised duration will be ignored as the tone will have already been terminated by the receiver. Note that it is not necessary to send **signalUpdate** if the total duration was indicated in **signal**.

rtp contains parameters needed to align the tone or hookflash with an RTP/UDP stream (see Recommendation H.323). In **signalUpdate**, this element only needs to be included if multiple **signal** messages have been issued specifying different **LogicalChannelNumbers** and it is necessary to indicate which **signal** is to be updated.

timestamp specifies, in terms of the RTP timestamp of the primary encoder on the associated audio channel, the time at which the tone or hookflash should be generated (delivered or injected into the audio stream). The tone or hookflash shall not be generated before audio with the same timestamp is played; it should be generated as soon as possible after this time, but not later than the **expirationTime** timestamp. The sender of an indication shall not set **timestamp** to a time that is "in the future"; **timestamp** is normally set equal to the timestamp of audio currently being sent or most recently sent on the associated audio channel. If **timestamp** is not specified, the signal shall be delivered or injected upon receipt.

expirationTime specifies, in terms of the RTP timestamp of the primary encoder on the associated audio channel, the time after which the tone or hookflash shall be considered "stale" and discarded by the receiver. Endpoints that receive **signal** and are unable to act on it before the **expirationTime** timestamp on the associated channel shall discard the message. If an **expirationTime** time is not specified by the sender, the message may nevertheless be discarded as a result of local configuration of the recipient.

logicalChannelNumber shall specify the **LogicalChannelNumber** of the associated audio channel, the context in which **timestamp** and **expirationTime** are meaningful.

An MC shall convert the timestamps and logical channel number from the received indication into the correct logical channel number and timestamps for each output channel when it forwards the indication to each receiving endpoint (the timestamps might change if the audio is being transcoded or mixed in an MP). An MC receiving an indication after the **expirationTime** time may discard the message immediately without forwarding it; otherwise, the MC shall forward all requests immediately without waiting for **timestamp** time to occur.

Endpoints shall use the **alphanumeric** indication to convey DTMF user input if the other endpoint has not indicated the ability to receive DTMF using **UserInputCapability**.

An endpoint which has the capability to receive DTMF indications using **signal** shall also be able to accept **alphanumeric** indications for compatibility with older terminals. An **alphanumeric**

indication may be treated as a sequence of one or more **signal** indications with **duration**, **timestamp** and **expirationTime** elements omitted, and characters not valid in **signalType** being discarded.

In typical usage, a gateway detecting DTMF in the audio stream from a PSTN channel will send **signal** immediately upon detection of a tone, using a relatively high estimate of **duration**, and begin measuring the tone duration. When the tone ends, **signalUpdate** is sent to indicate the total measured duration. If the tone has not ended but the measured duration is approaching the previous estimate (such that the estimate might be exceeded by the measured duration before a **signalUpdate** could be received), a **signalUpdate** is sent increasing the estimate. The frequency of sending **signalUpdate**, the initial duration estimate sent in **signal**, and the amount by which subsequent estimates are increased, are left to the implementer, but caution should be exercised so as to not burden the network with large numbers of **signalUpdate** messages and to avoid premature expiration of previous estimates.

In typical usage from a non-gateway endpoint, the **signal** element will contain the total duration of the tone to be produced by the gateway. In some applications, however, it may be desirable to provide real-time interactive control of tone duration to the user. In this case, **signal** and **signalUpdate** would be used in a manner similar to that described for gateways in the preceding paragraph, with **signal** being sent upon the activation of the user input (e.g. depression of a key or on-screen control) using an estimated tone duration, and **signalUpdate** used to send updated estimates so long as the input continues to be activated and to indicate the total duration when the input is deactivated.

B.13.7 Conference Indications

sbeNumber shall be defined as H.230 SBE Number.

terminalNumberAssign shall be defined as H.230 TIA.

terminalJoinedConference shall be defined as H.230 TIN.

terminalLeftConference shall be defined as H.230 TID.

seenByAtLeastOneOther shall be defined as H.230 MIV.

cancelSeenByAtLeastOneOther shall be defined as H.230 cancel-MIV.

seenByAll shall be defined as H.230 MIV.

cancelSeenByAll shall be defined as H.230 MIV.

terminalYouAreSeeing shall be defined as H.230 VIN.

requestForFloor shall be defined as H.230 TIF and be sent from a terminal to the MC.

withdrawChairToken shall be defined as H.230 CCR and is sent from the MC to the Chair Token holder.

floorRequested shall be defined as H.230 TIF when sent from the MC to the Chair Token holder. This request included the TerminalLabel of the requesting terminal.

B.13.8 H2250 Maximum Logical Channel Skew

H2250MaximumSkewIndication indicates the maximum skew between logical channels.

skew is measured in milliseconds, and indicates the maximum number of milliseconds that the data on logicalChannelNumber2 is delayed from the data on logicalChannelNumber1 as delivered to the network transport. The skew is measured relative to the time of delivery to the network transport of the first bit of data representing a given sample point. Lip synchronization, if desired, is a local matter for the receiver and shall be achieved via use of timestamps.

B.13.9 MC Location Indication

This indication is sent by the MC to indicate to other terminals the signalling address that should be used to reach the MC.

B.13.10 Vendor Identification Indication

vendorIdentification indication should be sent at the start of each call to identify the manufacturer, product, and product version number.

B.13.11 Function Not Supported

This is used to return requests, responses and commands that are not understood back to the transmitter.

The whole of the RequestMessage, ResponseMessage or CommandMessage is returned.

If a terminal receives a request, response or command that it does not understand, either because it is non-standard or has been defined in a subsequent revision of this Recommendation, it shall respond by sending FunctionNotSupported.

If a terminal receives a request, response or command that has incorrect encoding, it shall set cause to the value syntaxError. If it has correct encoding, but the encoded values are semantically incorrect, it shall set cause to the value semanticError. If the message is an unrecognized extension to MultimediaSystemControlMessage, RequestMessage, ResponseMessage or CommandMessage, it shall set cause to the value unknownFunction.

In each case, the whole MultimediaSystemControlMessage should be returned as an octet string in returnedFunction.

FunctionNotSupported shall not be used at any other time. In particular, when an unrecognized extension is present at other points in the syntax, FunctionNotSupported shall not be used: the terminal shall respond to the message in the normal way, as if no extension were present. FunctionNotSupported shall never be sent in response to a received indication.

ANNEX C

Procedures

C.1 Introduction

This annex defines generic multimedia system control procedures that use the messages defined in this Recommendation. Recommendations using this Recommendation shall indicate which of these procedures are applicable, as well as defining any specific requirements.

Procedures to perform the following functions are described in this clause:

- master-slave determination;
- terminal capability exchange;
- unidirectional logical channel signalling;
- bidirectional logical channel signalling;
- receive terminal close logical channel request;
- H.223 multiplex table entry modification;
- request multiplex entry;
- receiver to transmitter transmit mode request;

- round trip delay determination;
- maintenance loop.

C.1.1 Method of specification

Procedures are generally specified in this clause using SDLs. The SDL provides a graphical specification of the procedures, and includes specification of actions in the event of exception conditions.

C.1.2 Communication between protocol entity and protocol user

The interaction with the user of a particular function is specified in terms of primitives transferred at the interface between the protocol entity and the protocol user. Primitives are for the purpose of defining protocol procedures and are not intended to specify or constrain implementation. There may be a number of parameters associated with each primitive.

To assist in the specification, protocol states are defined. These states are conceptual and reflect general conditions of the protocol entity in the sequences of primitives exchanged between the protocol entity and the user, and the exchange of messages between the protocol entity and its peer.

For each protocol entity the allowed sequence of primitives between the user and the protocol entity is defined using a state transition diagram. The allowed sequence constrains the actions of the user, and defines the possible responses from the protocol entity.

A primitive parameter described as being null, is equivalent to the parameter not being present.

C.1.3 Peer-to-peer communication

Protocol information is transferred to the peer protocol entity via the relevant messages defined in clause 6. Some protocol entities described have state variables associated with them. A number of protocol entities described also have timers associated with them.

A timer is identified by the notation T_n , where n is a number. In the SDL diagrams setting a timer means that a timer is loaded with a specified value and the timer is started. Resetting a timer means that a timer is stopped with its value at the time of reset being retained. Timer expiry means that a timer has run for its specified time and has reached the value of zero.

A protocol entity may also have associated parameters. A parameter is identified by the notation N_n , where n is a number.

These timers and counters are listed in Appendix III.

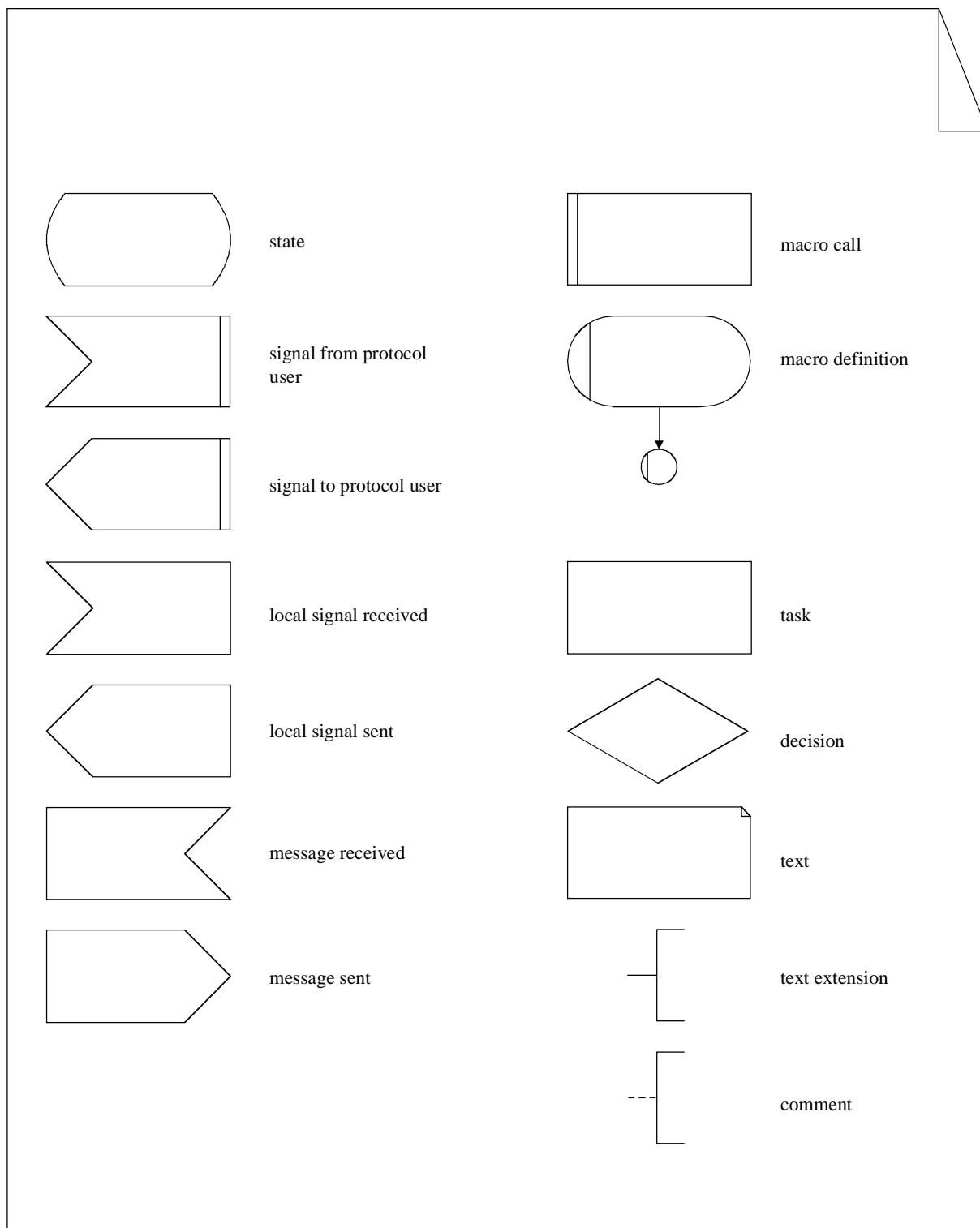
Some protocol entities define an error primitive to report protocol error conditions to a management entity.

C.1.4 SDL diagrams

The SDL diagrams show actions to the allowed interactions with the protocol user, and to reception of messages from the peer protocol entity. Primitives which are not allowed for a given state, as specified by the state transition diagrams, are not shown in the SDL diagrams. However, the responses to the reception of inappropriate messages are described in the SDL diagrams.

C.1.5 SDL Key

The SDL key is shown in Figure C.1.



T1519120-95

Figure C.1/H.245 – SDL key

C.2 Master slave determination procedures

C.2.1 Introduction

Conflicts may arise when two or more terminals involved in a call initiate similar events simultaneously, for which resources are available for only one occurrence of the event e.g. opening of logical channels. To resolve such conflicts, one terminal may act as a master and the other terminal(s) may act as slave terminal(s). The procedures described here allow terminals in the call to determine which is the master terminal and which is the slave terminal(s).

The protocol described here is referred to as the Master Slave Determination Signalling Entity (MSDSE). There is one instance of the MSDSE in each terminal involved in a call.

Either terminal may initiate the master slave determination process by issuing the DETERMINE.request primitive to its MSDSE. The result of the procedure is returned by the DETERMINE.indication and DETERMINE.confirm primitives. While the DETERMINE.indication primitive indicates the result, it does not indicate that the result is known at the remote terminal. The DETERMINE.confirm primitive indicates the result and confirms that it is also known at the remote terminal. A terminal may only initiate the master slave determination process if no procedure which depends upon its result is locally active.

A terminal shall respond to procedures that rely on knowledge of the result and are initiated by the remote terminal any time after the status determination result is known at the local terminal. This may be before the local terminal has received confirmation that the remote terminal also has knowledge of the result. A terminal shall not initiate procedures that rely on knowledge of the result until it has received confirmation that the remote terminal also has knowledge of the result of the current instance of the determination procedure.

The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

C.2.1.1 Protocol overview – Initiation by local user

A master slave determination procedure is initiated when the DETERMINE.request primitive is issued by the MSDSE user. A MasterSlaveDetermination message is sent to the peer MSDSE, and timer T106 is started. If a MasterSlaveDeterminationAck message is received in response to the MasterSlaveDetermination message, then timer T106 is stopped and the user is informed with the DETERMINE.confirm primitive that the master slave determination procedure was successful and a MasterSlaveDeterminationAck message is sent to the peer MSDSE. If however a MasterSlaveDeterminationReject message is received in response to the MasterSlaveDetermination message, then a new status determination number is generated, timer T106 is restarted, and another MasterSlaveDetermination message is sent. If after sending a MasterSlaveDetermination message N100 times, a MasterSlaveDeterminationAck still has not been received, then timer T106 is stopped and the user is informed with the REJECT.indication primitive that the master slave determination procedure has failed to produce a result.

If timer T106 expires then the MSDSE user is informed with the REJECT.indication primitive and a MasterSlaveDeterminationRelease message is sent to the peer MSDSE.

C.2.1.2 Protocol overview – Initiation by remote user

When a MasterSlaveDetermination message is received at the MSDSE, a status determination procedure is initiated. If the status determination procedure returns a determinate result, then the user is informed of the master slave determination result with the DETERMINE.indication primitive, a MasterSlaveDeterminationAck message is sent to the peer MSDSE, and timer T106 is started. If a

MasterSlaveDeterminationAck message is received in response to the MasterSlaveDeterminationAck message, then timer T106 is stopped and the user is informed with the DETERMINE.confirm primitive that the master slave determination procedure was successful.

If timer T106 expires then the MSDSE user is informed with the REJECT.indication primitive.

If however the status determination procedure returns an indeterminate result, then the MasterSlaveDeterminationReject message is sent to the peer MSDSE.

C.2.1.3 Protocol overview – Simultaneous initiation

When a MasterSlaveDetermination message is received at the MSDSE that itself has already initiated a status determination procedure, and is awaiting a MasterSlaveDeterminationAck or MasterSlaveDeterminationReject message, then a status determination procedure is initiated. If the status determination procedure returns a determinate result, the MSDSE responds as if the procedure was initiated by the remote user, and the procedures described above for this condition apply.

If however the status determination procedure returns an indeterminate result, then a new status determination number is generated, and the MSDSE responds as if the procedure was again initiated by the local MSDSE user as described above.

C.2.1.4 Status determination procedure

The following procedure is used to determine which terminal is the master from the terminalType and statusDeterminationNumber values. Firstly, the terminalType values are compared and the terminal with the larger terminal type number is determined as the master. If the terminal type numbers are the same, the statusDeterminationNumbers are compared using modulo arithmetic to determine which is master.

If both terminals have equal terminalType field values and the difference between statusDeterminationNumber field values modulo 2^{24} is 0 or 2^{23} , an indeterminate result is obtained.

C.2.2 Communication between the MSDSE and the MSDSE user

C.2.2.1 Primitives between the MSDSE and the MSDSE user

Communication between the MSDSE, and MSDSE user, is performed using the primitives shown in Table C.1.

Table C.1/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
DETERMINE	– (Note 1)	TYPE	not defined (Note 2)	TYPE
REJECT	not defined	–	not defined	not defined
ERROR	not defined	ERRCODE	not defined	not defined
NOTE 1 – "–" means no parameters.				
NOTE 2 – "not defined" means that this primitive is not defined.				

C.2.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The DETERMINE primitive is used to initiate, and to return the result from, the master slave determination procedure.

The DETERMINE.request primitive is used to initiate the master slave determination procedure.

The DETERMINE.indication primitive is used to indicate the result of the master slave determination procedure. As the result of the procedure may not be known at the remote terminal, the terminal shall not initiate any procedures that rely on knowledge of the result, although it shall respond to any procedures that rely on knowledge of the result.

The DETERMINE.confirm primitive is used to indicate the result of the master slave determination procedure and that the result of the procedure is known at both terminals. The terminal may initiate, and shall respond to, any procedures that rely on knowledge of the result.

- b) The REJECT primitive indicates that the master slave determination procedure was unsuccessful.
- c) The ERROR primitive reports MSDSE errors to a management entity.

C.2.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.1 is as follows:

- a) The TYPE parameter indicates the terminal status. It has the value of "MASTER" or "SLAVE".
- b) The ERRCODE value indicates the type of MSDSE error Table C.5 indicates the values that the ERRCODE parameter may take.

C.2.2.4 MSDSE states

The following states are used to specify the allowed sequence of primitives between the MSDSE and the MSDSE user.

State 0: IDLE

No master slave determination procedure has been initiated.

State 1: OUTGOING AWAITING RESPONSE

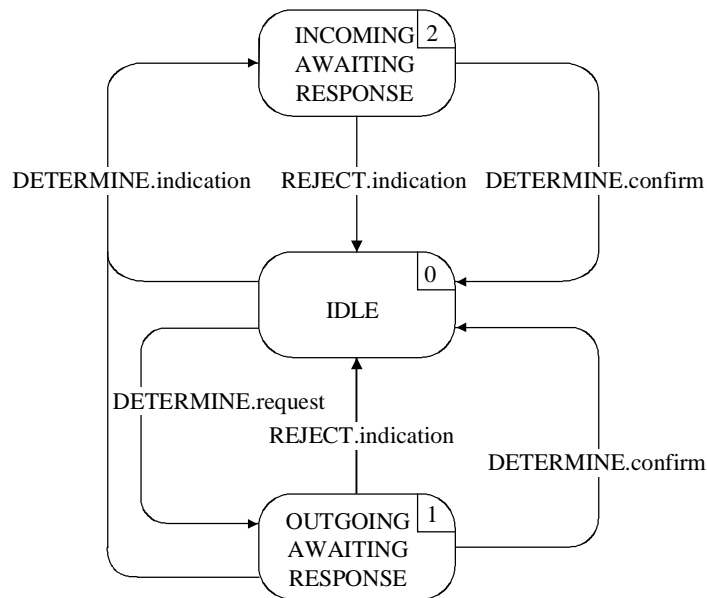
The local MSDSE user has requested a master slave determination procedure. A response from the remote MSDSE is awaited.

State 2: INCOMING AWAITING RESPONSE

The remote MSDSE has initiated a master slave determination procedure in the local MSDSE. An acknowledgement was sent to the remote MSDSE and a response from the remote MSDSE is awaited.

C.2.2.5 State transition diagram

The allowed sequence of primitives between the MSDSE and the MSDSE user is defined here. The allowed sequences are shown in Figure C.2.



T1520240-95

Figure C.2/H.245 – State transition diagram for sequence of primitives at MSDSE

C.2.3 Peer-to-peer MSDSE communication

C.2.3.1 MSDSE messages

Table C.2 shows the MSDSE messages and fields, defined in clause 6, which are relevant to the MSDSE protocol.

Table C.2/H.245 – MSDSE message names and fields

Function	Message	Field
determination	MasterSlaveDetermination	terminalType statusDeterminationNumber
	MasterSlaveDeterminationAck	decision
	MasterSlaveDeterminationReject	cause
error recovery	MasterSlaveDeterminationRelease	–

C.2.3.2 MSDSE state variables

The following MSDSE state variables are defined:

- sv_TT
This state variable holds the terminal type number for this terminal.
- sv_SDNUM
This state variable holds the status determination number for this terminal.
- sv_STATUS
This state variable is used to store the result of the latest master slave determination procedure. It has values of "master", "slave", and "indeterminate".

- sv_NCOUNT
This state variable is used to count the number of MasterSlaveDetermination messages that have been sent during the OUTGOING AWAITING RESPONSE state.

C.2.3.3 MSDSE timers

The following timer is specified for the outgoing MSDSE:

- T106
This timer is used during the OUTGOING AWAITING RESPONSE state and during the INCOMING AWAITING RESPONSE state. It specifies the maximum allowed time during which no acknowledgement message may be received.

C.2.3.4 MSDSE counters

The following parameter is specified for the MSDSE:

- N100
This parameter specifies the maximum value of sv_NCOUNT.

C.2.4 MSDSE procedures

C.2.4.1 Introduction

Figure C.3 summarizes the MSDSE primitives and their parameters, and messages.

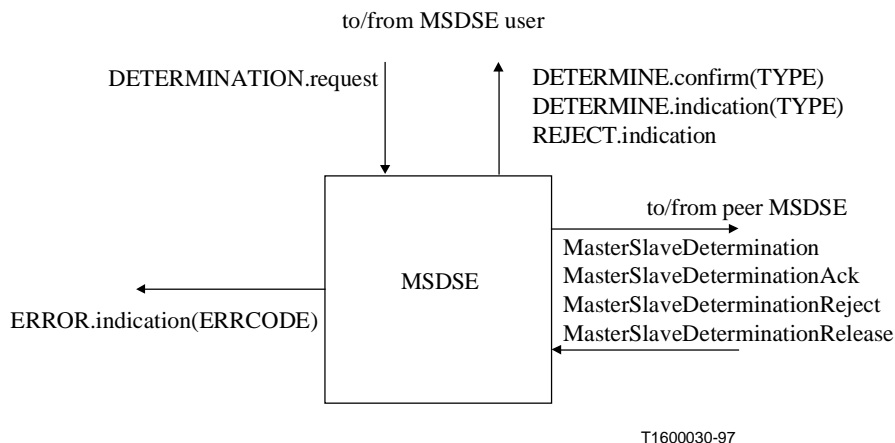


Figure C.3/H.245 – Primitives and messages in the MSDSE

C.2.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams, the parameters of the indication and confirm primitives assume values as shown in Table C.3.

Table C.3/H.245 – Default primitive parameter values

Primitive	Parameter	Default value
DETERMINE.confirm	TYPE	MasterSlaveDeterminationAck.decision
DETERMINE.indication	TYPE	sv_STATUS

C.2.4.3 Message field default values

Where not explicitly stated in the SDL diagrams, the message fields assume values as shown in Table C.4.

Table C.4/H.245 – Default message field values

Message	Field	Default value
MasterSlaveDetermination	terminalType	sv_TT
	statusDeterminationNumber	sv_SDNUM
MasterSlaveDeterminationAck	decision	Opposite of sv_STATUS i.e. if (sv_STATUS == master) decision = slave if (sv_STATUS == slave) decision = master
MasterSlaveDeterminationReject	cause	identicalNumbers

C.2.4.4 ERRCODE parameter values

Table C.5 shows the values that the ERRCODE parameter of the ERROR.indication primitive may take for the MSDSE.

Table C.5/H.245 – ERRCODE parameter values at MSDSE

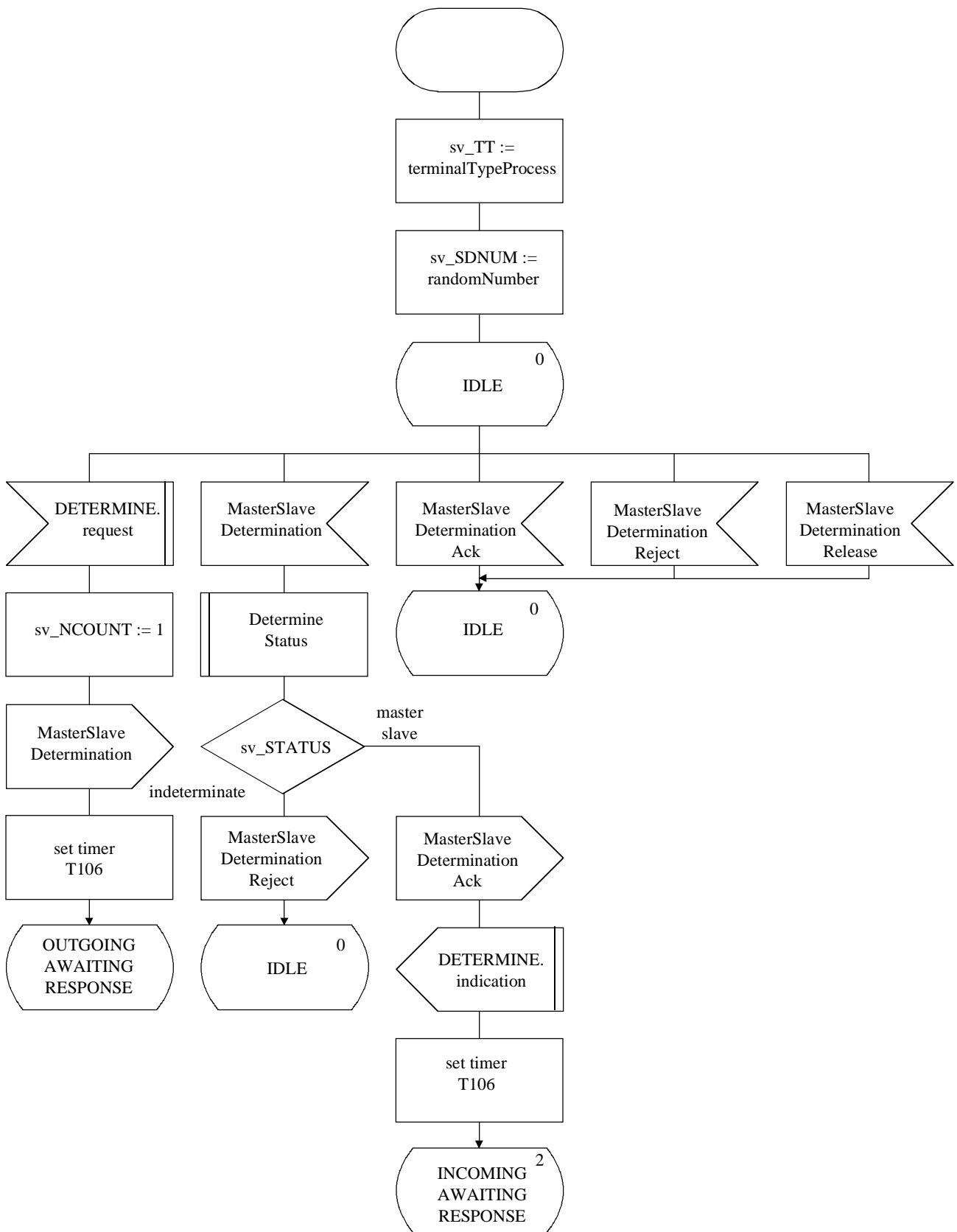
Error type	Error code	Error condition	State
No response from remote MSDSE	A	Local timer T106 expiry	OUTGOING AWAITING RESPONSE INCOMING AWAITING RESPONSE
Remote sees no response from local MSDSE	B	Remote timer T106 expiry	OUTGOING AWAITING RESPONSE INCOMING AWAITING RESPONSE
Inappropriate message	C	MasterSlaveDetermination	INCOMING AWAITING RESPONSE
	D	MasterSlaveDetermination Reject	INCOMING AWAITING RESPONSE
Inconsistent field value	E	MasterSlaveDetermination Ack.decision != sv_STATUS	INCOMING AWAITING RESPONSE
Maximum number of retries	F	sv_NCOUNT == N100	OUTGOING AWAITING RESPONSE

C.2.4.5 SDLs

The MSDSE procedures are expressed in SDL form in Figure C.4.

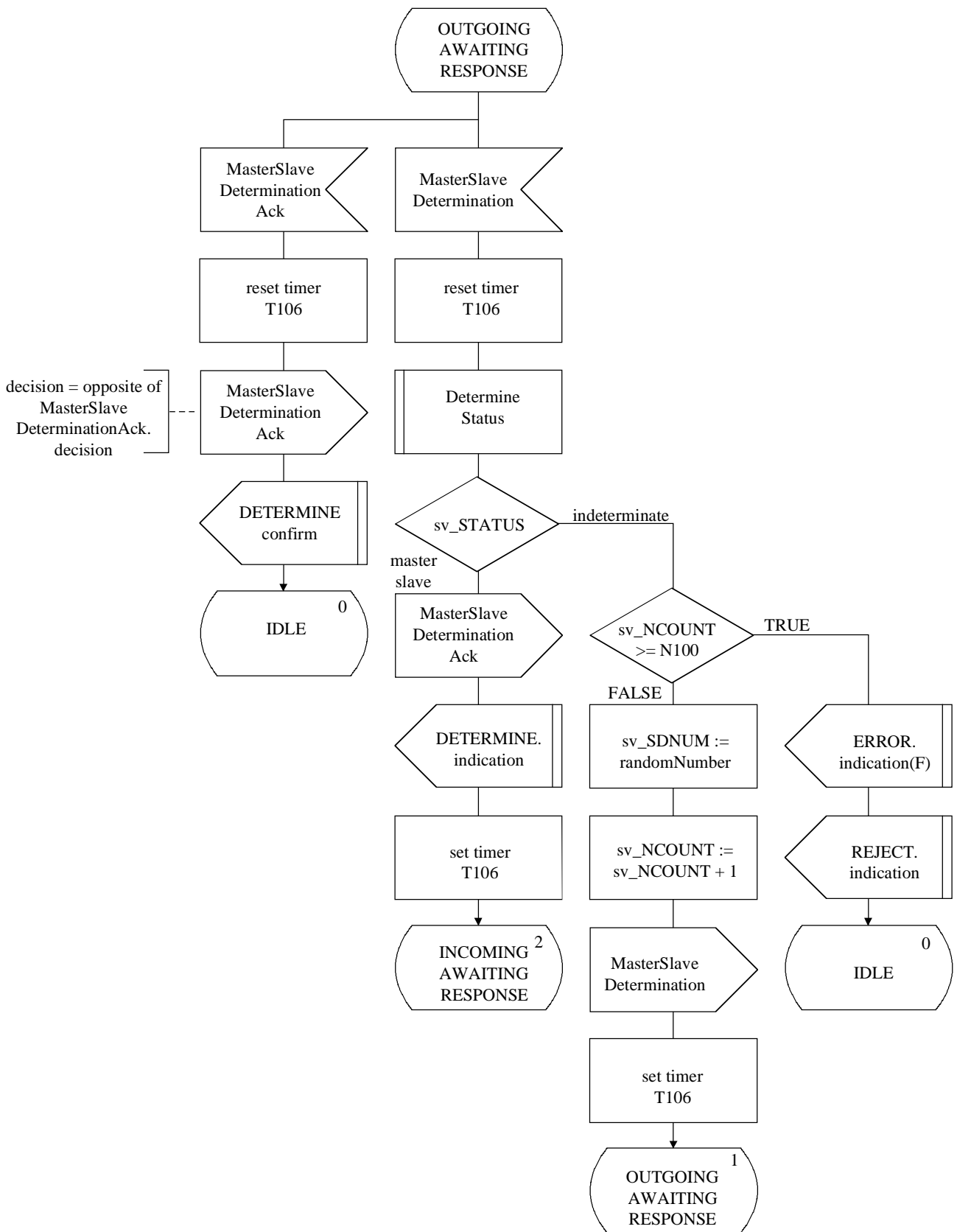
terminalTypeProcess is process that returns a number that identifies different types of terminal, such as, terminals, MCUs and gateways.

randomNumber is process that returns a random number in the range $0 \dots 2^{24} - 1$.



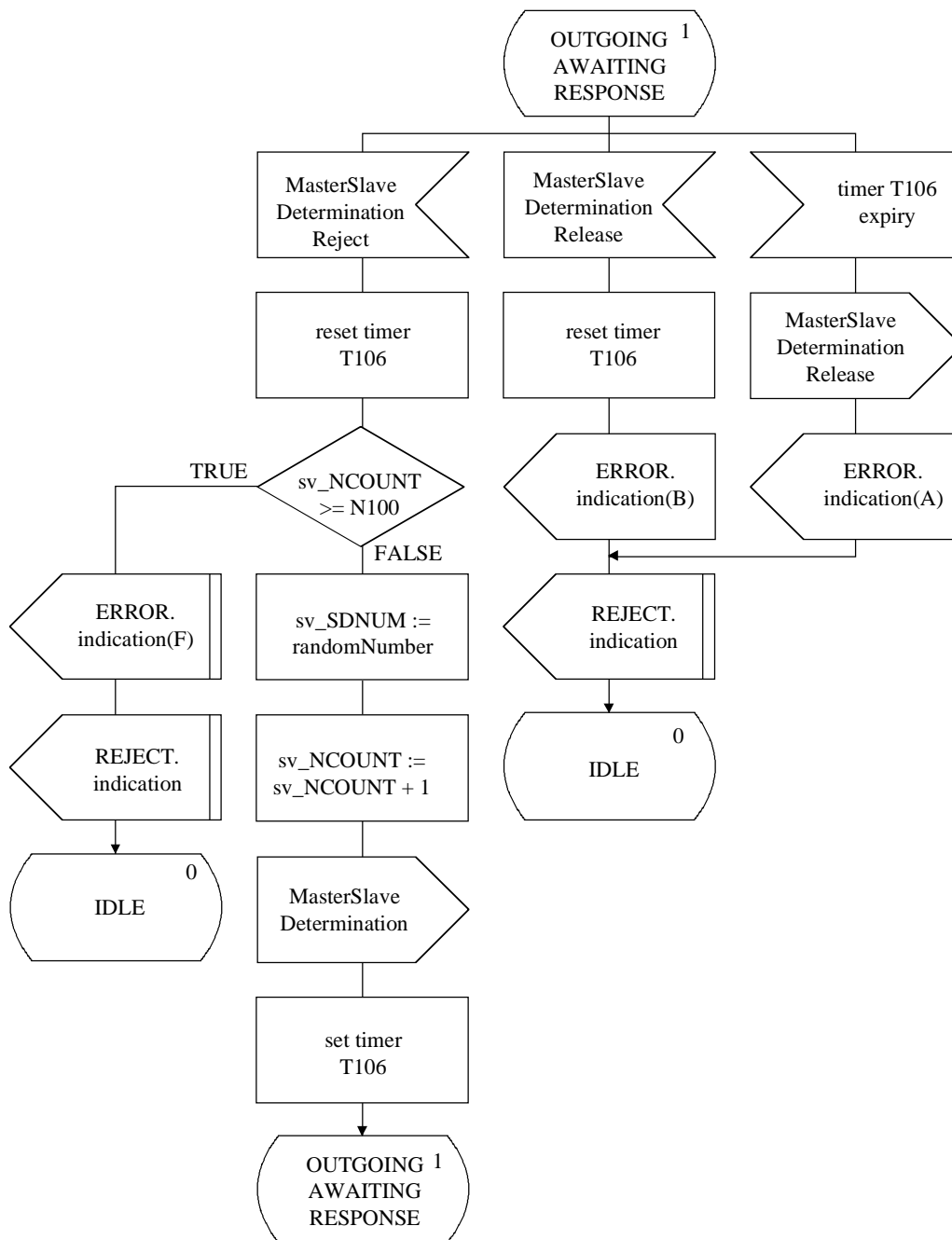
T1600040-97

Figure C.4/H.245 – MSDSE SDL (sheet 1 of 5)



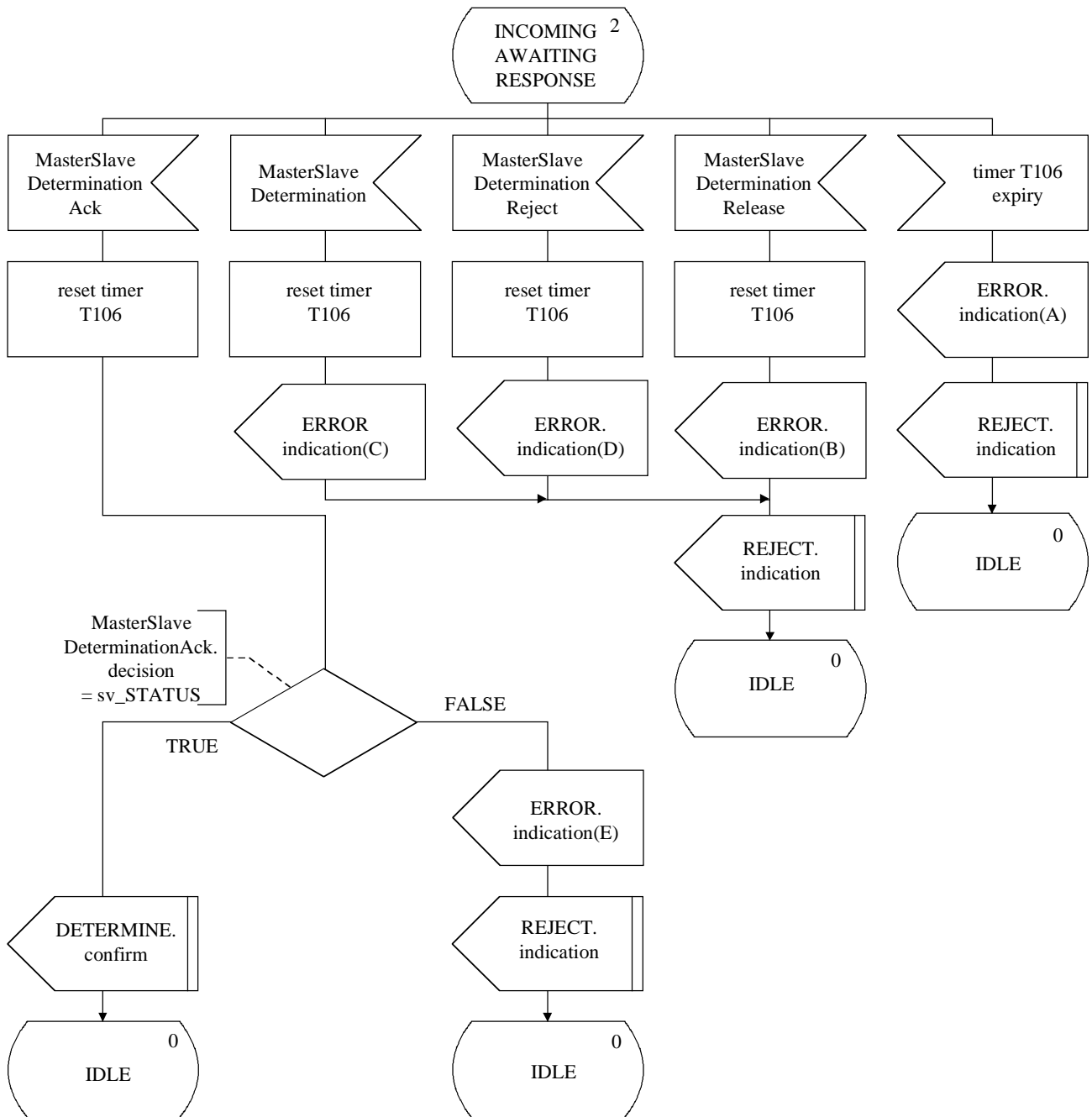
T1520260-95

Figure C.4/H.245 – MSDSE SDL (sheet 2 of 5)



T1520270-95

Figure C.4/H.245 – MSDSE SDL (sheet 3 of 5)



T1520280-95

Figure C.4/H.245 – MSDSE SDL (sheet 4 of 5)

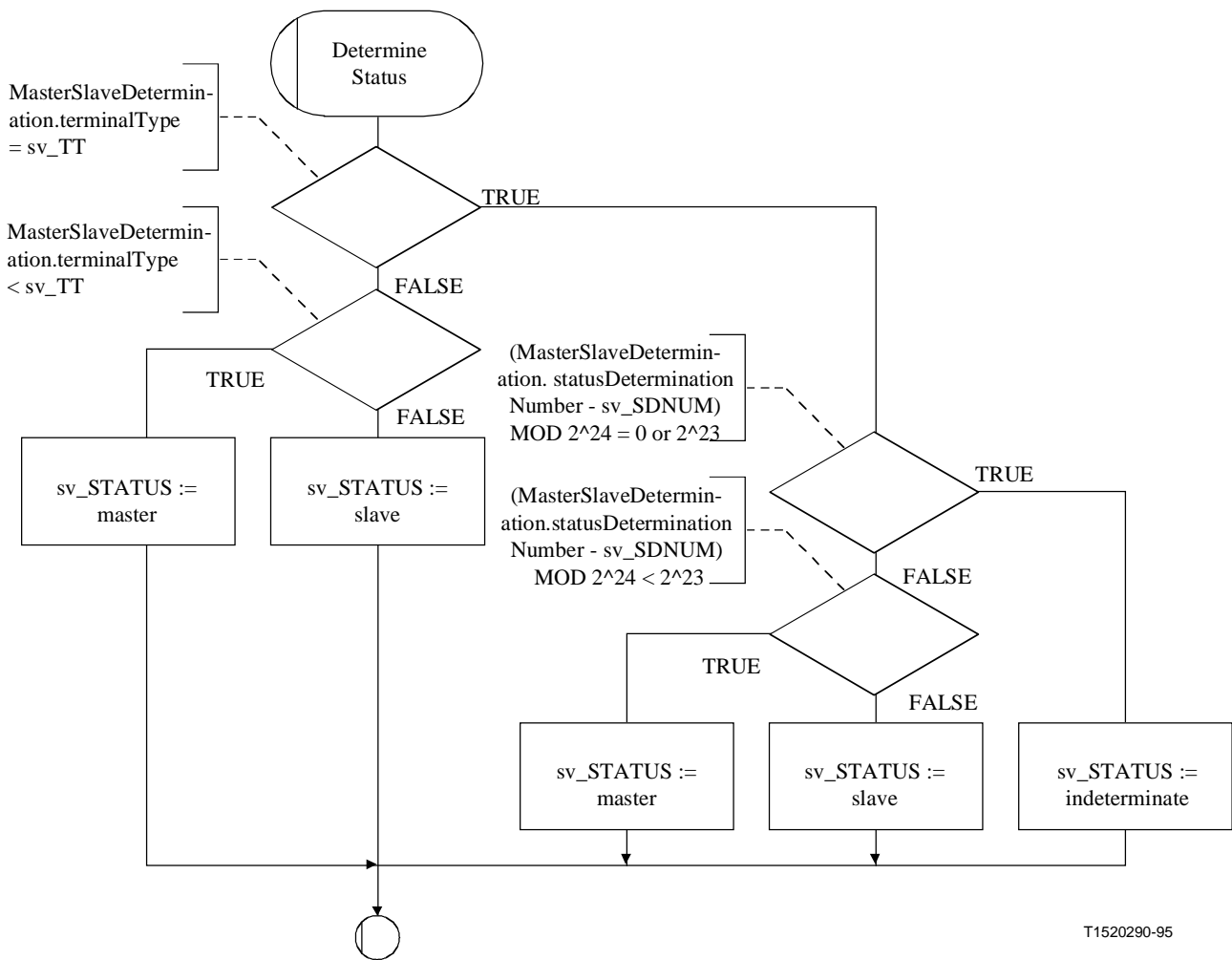


Figure C.4/H.245 – MSDSE SDL (sheet 5 of 5)

C.3 Capability exchange procedures

C.3.1 Introduction

These procedures are used by terminals to communicate their capabilities, and are referred to as the Capability Exchange Signalling Entity (CESE). Procedures are specified in terms of primitives and states at the interface between the CESE and the CESE user. Protocol information is transferred to the peer CESE via relevant messages defined in clause 6. There is an outgoing CESE and an incoming CESE. At each of the outgoing and incoming ends there is one instance of the CESE for each call.

All terminals intended for use in point-to-point applications or those connected to an MCU shall be able to identify a TerminalCapabilitySet and its structure, and such capability values therein that are mandatory for those applications; any unrecognised capability values shall be ignored, and no fault shall be implied.

The capability exchange may be performed at any time. The capability exchange may signal both changed and unchanged capabilities. Unchanged capabilities should not be sent repetitively without strong cause.

The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

C.3.1.1 Protocol overview – Outgoing CESE

A capability exchange is initiated when the TRANSFER.request primitive is issued by the user at the outgoing CESE. A TerminalCapabilitySet message is sent to the peer incoming CESE, and timer T101 is started. If a TerminalCapabilitySetAck message is received in response to the TerminalCapabilitySet message then timer T101 is stopped and the user is informed with the TRANSFER.confirm primitive that the capability exchange was successful. If however a TerminalCapabilitySetReject message is received in response to the TerminalCapabilitySet message then timer T101 is stopped and the user is informed with the REJECT.indication primitive that the peer CESE user has refused the capability exchange.

If timer T101 expires then the outgoing CESE user is informed with the REJECT.indication primitive and a TerminalCapabilitySetRelease message is sent.

C.3.1.2 Protocol overview – Incoming CESE

When a TerminalCapabilitySet message is received at the incoming CESE, the user is informed of the capability exchange request with the TRANSFER.indication primitive. The incoming CESE user signals acceptance of the capability exchange request by issuing the TRANSFER.response primitive, and a TerminalCapabilitySetAck message is sent to the peer outgoing CESE. The incoming CESE user signals rejection of the capability exchange request by issuing the REJECT.request primitive, and a TerminalCapabilitySetReject message is sent to the peer outgoing CESE.

C.3.2 Communication between CESE and CESE user

C.3.2.1 Primitives between CESE and CESE user

Communication between the CESE and CESE user, is performed using the primitives shown in Table C.6.

Table C.6/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
TRANSFER	PROTOID MUXCAP CAPTABLE CAPDESCRIPTORS	PROTOID MUXCAP CAPTABLE CAPDESCRIPTORS	– (Note 1)	–
REJECT	CAUSE	SOURCE CAUSE	not defined (Note 2)	not defined
NOTE 1 – "–" means no parameters.				
NOTE 2 – "not defined" means that this primitive is not defined.				

C.3.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The TRANSFER primitives are used for transfer of the capability exchange.
- b) The REJECT primitives are used to reject a capability descriptor entry, and to terminate a current capability transfer.

C.3.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.6 is as follows:

- a) The PROTOID parameter is the protocol identifier parameter. This parameter is mapped to the protocolIdentifier field of the TerminalCapabilitySet message and carried transparently to the peer CESE user. This parameter is mandatory.
- b) The MUXCAP parameter is the multiplex capability parameter. This parameter is mapped to the multiplexCapability field of the TerminalCapabilitySet message and carried transparently to the peer CESE user. This parameter is optional.
- c) The CAPTABLE parameter is the capability table parameter. There may be one or more capability table entries described within this parameter. This parameter is mapped to the capabilityTable field of the TerminalCapabilitySet message and carried transparently to the peer CESE user. This parameter is optional.
- d) The CAPDESCRIPTORS parameter is the capability descriptors parameter. There may be one or more capability descriptors described within in this parameter. This parameter is mapped to the capabilityDescriptors field of the TerminalCapabilitySet message and carried transparently to the peer CESE user. This parameter is optional.
- e) The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of "USER" or "PROTOCOL". The latter case may occur as the result of a timer expiry.
- f) The CAUSE parameter indicates the reason for rejection of a CAPTABLE or CAPDESCRIPTORS parameter. The CAUSE parameter is not present when the SOURCE parameter indicates "PROTOCOL".

C.3.2.4 CESE states

The following states are used to specify the allowed sequence of primitives between the CESE and the CESE user.

The states for an outgoing CESE are:

State 0: IDLE

The CESE is idle.

State 1: AWAITING RESPONSE

The CESE is waiting for a response from the remote CESE.

The states for an incoming CESE are:

State 0: IDLE

The CESE is idle.

State 1: AWAITING RESPONSE

The CESE is waiting for a response from the CESE user.

C.3.2.5 State transition diagram

The allowed sequence of primitives between the CESE and the CESE user is defined here. The allowed sequence of primitives relates to states of the CESE as viewed from the CESE user. The allowed sequences are specified separately for each of an outgoing CESE and an incoming CESE, as shown in Figure C.5 and Figure C.6 respectively.

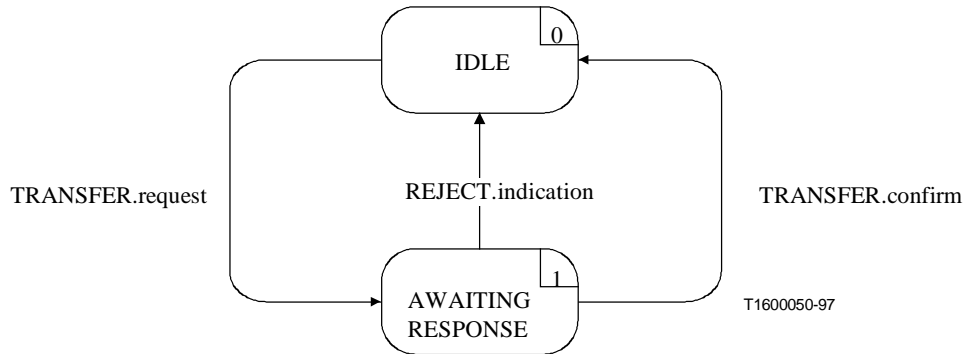


Figure C.5/H.245 – State transition diagram for sequence of primitives at CESE outgoing

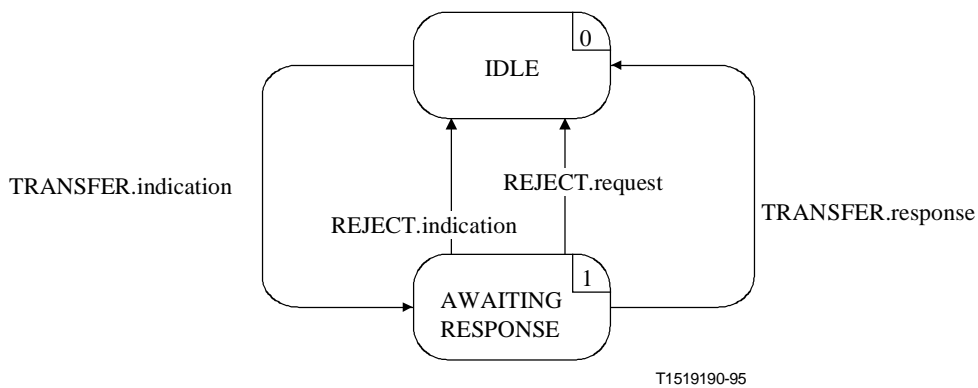


Figure C.6/H.245 – State transition diagram for sequence of primitives at CESE incoming

C.3.3 Peer-to-peer CESE communication

C.3.3.1 Messages

Table C.7 shows the CESE messages and fields, defined in clause 6, which are relevant to the CESE protocol.

Table C.7/H.245 – CESE message names and fields

Function	Message	Direction	Field
transfer	TerminalCapabilitySet	O → I (Note)	sequenceNumber protocolIdentifier multiplexCapability capabilityTable capabilityDescriptors
	TerminalCapabilitySetAck	O ← I	sequenceNumber
reject	TerminalCapabilitySetReject	O ← I	sequenceNumber cause
reset	TerminalCapabilitySetRelease	O → I	–

NOTE – direction: O – outgoing, I – incoming.

C.3.3.2 CESE state variables

The following state variables are defined at the outgoing CESE:

out_SQ

This state variable is used to indicate the most recent TerminalCapabilitySet message. It is incremented by one and mapped to the TerminalCapabilitySet message sequenceNumber field before transmission of the TerminalCapabilitySet message. Arithmetic performed on out_SQ is modulo 256.

The following state variables are defined at the incoming CESE:

in_SQ

This state variable is used to store the value of the sequenceNumber field of the most recently received TerminalCapabilitySet message. The TerminalCapabilitySetAck and TerminalCapabilitySetReject messages have their sequenceNumber fields set to the value of in_SQ, before being sent to the peer CESE.

C.3.3.3 CESE timers

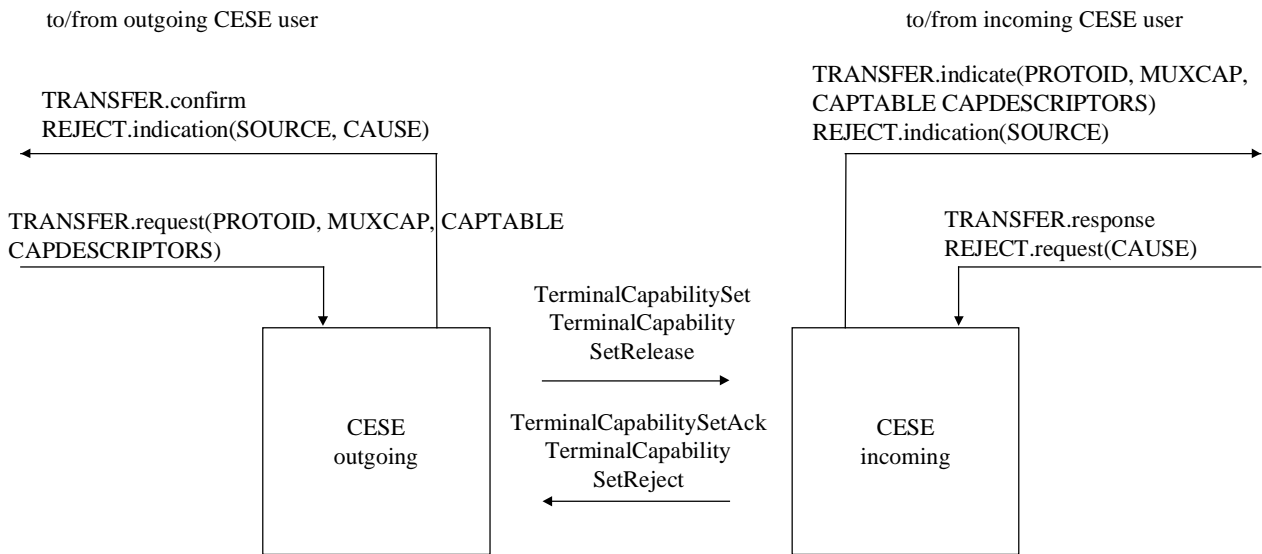
The following timer is specified for the outgoing CESE:

T101

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no TerminalCapabilitySetAck or TerminalCapabilitySetReject message may be received.

C.3.4 CESE procedures

Figure C.7 summarizes the CESE primitives and their parameters, and messages, for each of the outgoing and incoming CESE.



T1519200-95

Figure C.7/H.245 – Primitives and messages in the Capability Exchange Signalling Entity

C.3.4.1 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.8.

Table C.8/H.245 – Default primitive parameter values

Primitive	Parameter	Default value
TRANSFER.indication	PROTOID	TerminalCapabilitySet.protocolIdentifier
	MUXCAP	TerminalCapabilitySet.multiplexCapability
	CAPTABLE	TerminalCapabilitySet.capabilityTable
	CAPDESCRIPTORS	TerminalCapabilitySet.capabilityDescriptors
REJECT.indication	SOURCE	USER
	CAUSE	null

C.3.4.2 Message field default values

Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.9.

Table C.9/H.245 – Default message field values

Message	Field	Default value (Note)
TerminalCapabilitySet	sequenceNumber protocolIdentifier multiplexCapability capabilityTable capabilityDescriptors	out_SQ TRANSFER.request(PROTOID) TRANSFER.request(MUXCAP) TRANSFER.request(CAPTABLE) TRANSFER.request(CAPDESCRIPTORS)
TerminalCapabilitySetAck	sequenceNumber	in_SQ
TerminalCapabilitySetReject	sequenceNumber cause	in_SQ REJECT.request(CAUSE)
TerminalCapabilitySetRelease	–	–
NOTE – A message field shall not be coded, if the corresponding primitive parameter is null i.e. not present.		

C.3.4.3 SDLs

The outgoing CESE and the incoming CESE procedures are expressed in SDL form in Figure C.8 and Figure C.9, respectively.

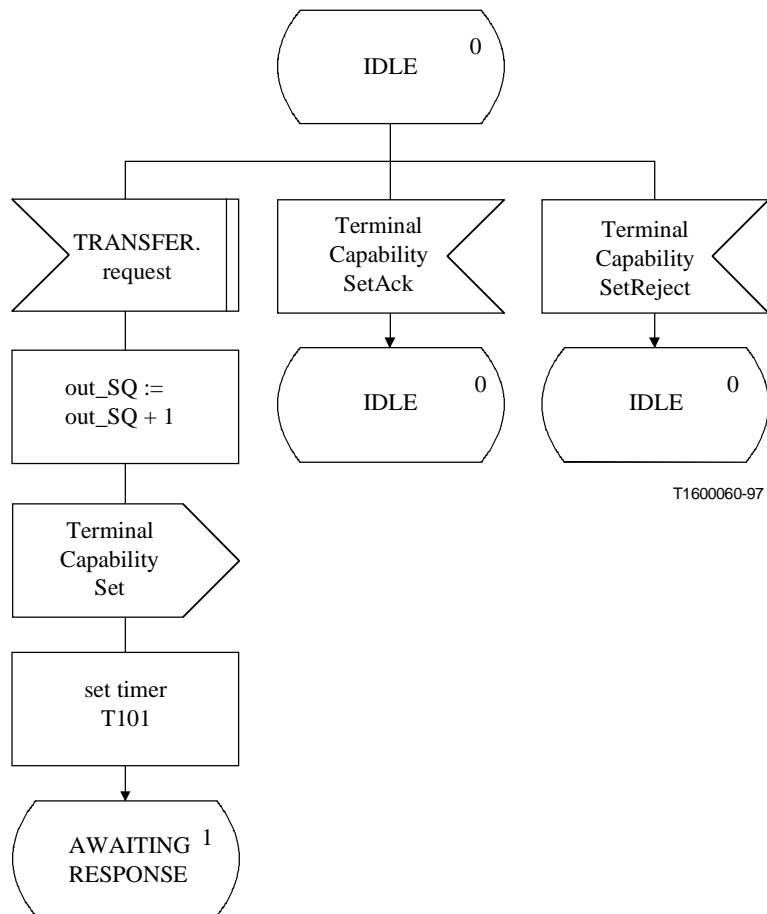
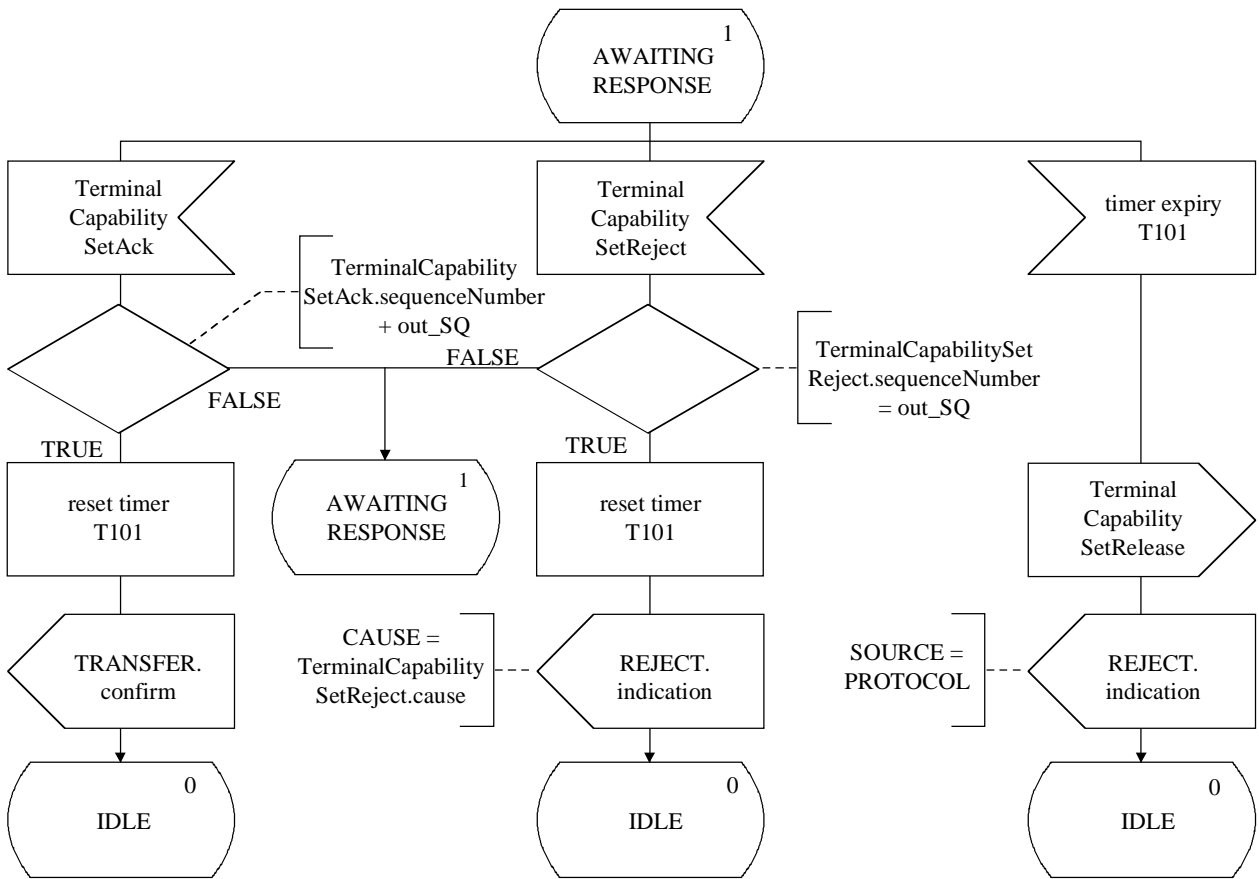
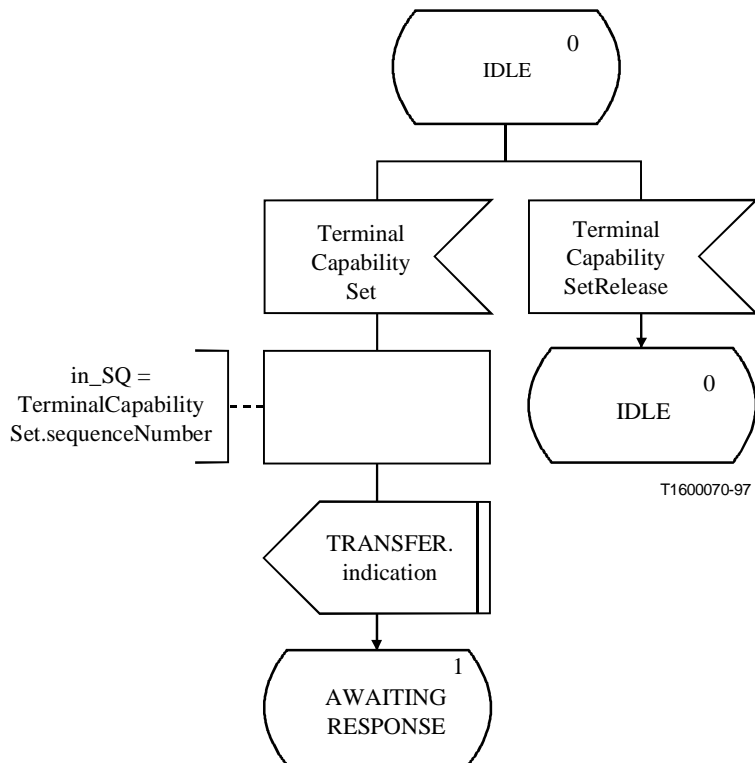


Figure C.8/H.245 – Outgoing CESE SDL (sheet 1 of 2)



T1520300-95

Figure C.8/H.245 – Outgoing CESE SDL (sheet 2 of 2)



T1600070-97

Figure C.9 /H.245 – Incoming CESE SDL (sheet 1 of 2)

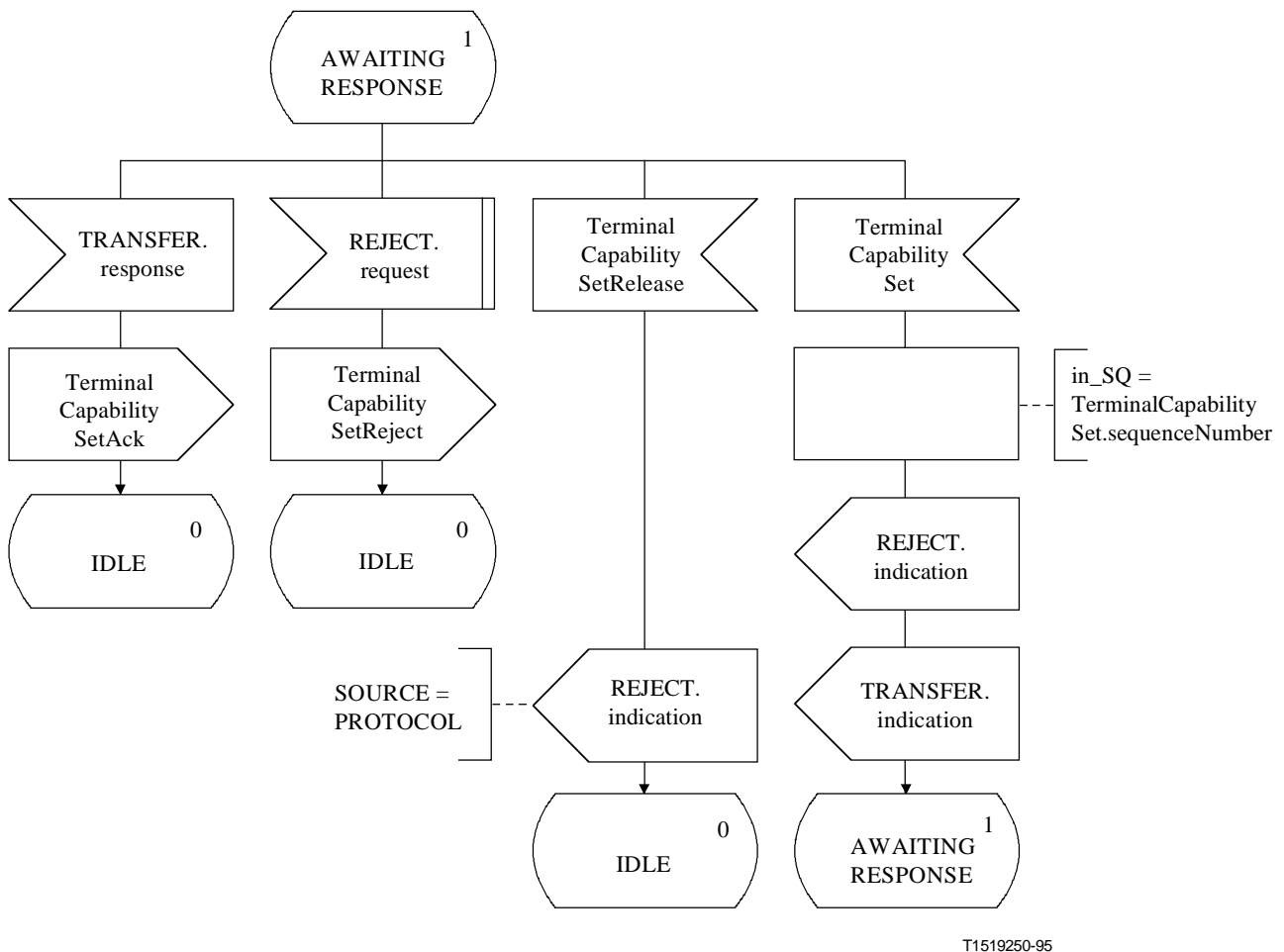


Figure C.9 /H.245 – Incoming CESE SDL (sheet 2 of 2)

C.4 Unidirectional Logical Channel signalling procedures

C.4.1 Introduction

The protocol specified here provides reliable opening and closing of unidirectional logical channels using acknowledged procedures.

The protocol specified here is referred to as the Logical Channel Signalling Entity (LCSE). Procedures are specified in terms of primitives at the interface between the LCSE and the LCSE user, and LCSE states. Protocol information is transferred to the peer LCSE via relevant messages defined in clause 6.

There is an outgoing LCSE and an incoming LCSE. At each of the outgoing and incoming sides there is one instance of the LCSE for each unidirectional logical channel. There is no connection between an incoming LCSE and an outgoing LCSE at one side, other than via primitives to and from the LCSE user. LCSE error conditions are reported.

Data shall only be sent on a logical channel in the ESTABLISHED state. If data is received on a logical channel that is not in the ESTABLISHED state the data shall be discarded and no fault shall be considered to have occurred.

Mode switching should be performed by closing and opening existing logical channels, or by opening new logical channels.

NOTE – Some Recommendations that use this Recommendation may define some default logical channels. These shall be considered ESTABLISHED from the start of communication and shall not be opened using these procedures. They may, however, be closed by these procedures, and subsequently be reopened for the same or a different purpose.

A terminal that is no longer capable of processing the signals on a logical channel should take appropriate action: this should include closing the logical channel and transmitting the relevant (changed) capability information to the remote terminal.

The following text provides an overview of the operation of the LCSE protocol. In the case of discrepancy between this and the formal specification, the formal specification will supersede.

C.4.1.1 Protocol overview

The opening of a logical channel is initiated when the ESTABLISH.request primitive is issued by the user at the outgoing LCSE. An OpenLogicalChannel message, containing forward logical channel parameters but not including reverse logical channel parameters, is sent to the peer incoming LCSE, and timer T103 is started. If an OpenLogicalChannelAck message is received in response to the OpenLogicalChannel message then timer T103 is stopped and the user is informed with the ESTABLISH.confirm primitive that the logical channel has been successfully opened. The logical channel may now be used to transmit user information. If however an OpenLogicalChannelReject message is received in response to the OpenLogicalChannel message then timer T103 is stopped and the user is informed with the RELEASE.indication primitive that the peer LCSE user has refused establishment of the logical channel.

If timer T103 expires in this period then the user is informed with the RELEASE.indication primitive, and a CloseLogicalChannel message is sent to the peer incoming LCSE.

A logical channel which has been successfully established may be closed when the RELEASE.request primitive is issued by the user at the outgoing LCSE. A CloseLogicalChannel message is sent to the peer incoming LCSE, and the timer T103 is started. When a CloseLogicalChannelAck message is received, timer T103 is stopped and the user is informed that the logical channel has been successfully closed with the RELEASE.confirm primitive.

If timer T103 expires in this period then the user is informed with the RELEASE.indication primitive.

Before either of the OpenLogicalChannelAck or OpenLogicalChannelReject messages have been received in response to a previously sent OpenLogicalChannel message, the user at the outgoing LCSE may close the logical channel using the RELEASE.request primitive.

Before the CloseLogicalChannelAck message is received in response to a previously sent CloseLogicalChannel message, the user at the outgoing LCSE may establish a new logical channel by issuing the ESTABLISH.request primitive.

C.4.1.2 Protocol overview – Incoming LCSE

When an OpenLogicalChannel message is received at the incoming LCSE, the user is informed of the request to open a new logical channel with the ESTABLISH.indication primitive. The incoming LCSE user signals acceptance of the request to establish the logical channel by issuing the ESTABLISH.response primitive, and an OpenLogicalChannelAck message is sent to the peer outgoing LCSE. The logical channel may now be used to receive user information. The incoming LCSE user signals rejection of the request to establish the logical channel by issuing the RELEASE.request primitive, and an OpenLogicalChannelReject message is sent to the peer outgoing LCSE.

A logical channel which has been successfully established may be closed when the CloseLogicalChannel message is received at the incoming LCSE. The incoming LCSE user is informed with the RELEASE.indication primitive, and the CloseLogicalChannelAck message is sent to the peer outgoing LCSE.

C.4.1.3 Conflict resolution

Conflicts may arise when requests to open logical channels are initiated at the same time. It may be possible to determine that there is a conflict from knowledge of exchanged capabilities.

Terminals shall be capable of detecting when conflict has arisen, or might arise, and shall act as follows:

Before logical channels can be opened, one terminal must be determined as the master terminal, and the other as the slave. The protocol defined in C.2 provides one means to make this decision. The master terminal shall reject immediately any request from the slave that it identifies as a conflicting request. The slave terminal may identify such conflicts, but shall respond to the request from the master terminal, with the knowledge that its earlier request will be rejected.

NOTE – Such conflicts might be caused by limited terminal resources, for example, when receive and transmission capabilities are dependent, as in the case of a terminal that can support a number of audio algorithms, but can only decode the same algorithm as it is encoding.

C.4.2 Communication between the LCSE and the LCSE user

C.4.2.1 Primitives between the LCSE and the LCSE user

Communication between the LCSE and the LCSE user is performed using the primitives shown in Table C.10.

Table C.10/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
ESTABLISH	FORWARD_PARAM	FORWARD_PARAM	– (Note 1)	–
RELEASE	CAUSE	SOURCE CAUSE	not defined (Note 2)	–
ERROR	not defined	ERRCODE	not defined	not defined
NOTE 1 – "-" means no parameters.				
NOTE 2 – "not defined" means that this primitive does not exist.				

C.4.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The ESTABLISH primitives are used to establish a logical channel for audiovisual and data communication.
- b) The RELEASE primitives are used to release a logical channel.
- c) The ERROR primitive reports LCSE errors to a management entity.

C.4.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.10 is as follows:

- a) The FORWARD_PARAM parameter specifies the parameters associated with the logical channel. This parameter is mapped to the forwardLogicalChannelParameters field of the OpenLogicalChannel message and is carried transparently to the peer LCSE user.
- b) The SOURCE parameter indicates to the LCSE user the source of the logical channel release. The SOURCE parameter has the value of "USER" or "LCSE", indicating either the LCSE user, or the LCSE. The latter may occur as the result of a protocol error.
- c) The CAUSE parameter indicates the reason as to why the peer LCSE user rejected a request to establish a logical channel. The CAUSE parameter is not present when the SOURCE parameter indicates "LCSE".
- d) The ERRCODE parameter indicates the type of LCSE error. Table C.14 shows the allowed values of the ERRCODE parameter.

C.4.2.4 LCSE states

The following states are used to specify the allowed sequence of primitives between the LCSE and the LCSE user, and the exchange of messages between peer LCSEs. The states are specified separately for each of an outgoing LCSE and an incoming LCSE. The states for an outgoing LCSE are:

State 0: RELEASED

The logical channel is released. The logical channel shall not be used to send outgoing data.

State 1: AWAITING ESTABLISHMENT

The outgoing LCSE is waiting to establish a logical channel with a peer incoming LCSE. The logical channel shall not be used to send outgoing data.

State 2: ESTABLISHED

The LCSE peer-to-peer logical channel connection has been established. The logical channel may be used to send outgoing data.

State 3: AWAITING RELEASE

The outgoing LCSE is waiting to release a logical channel with the peer incoming LCSE. The logical channel shall not be used to send outgoing data.

The states for an incoming LCSE are:

State 0: RELEASED

The logical channel is released. The logical channel shall not be used to receive incoming data.

State 1: AWAITING ESTABLISHMENT

The incoming LCSE is waiting to establish a logical channel with a peer outgoing LCSE. The logical channel shall not be used to receive incoming data.

State 2: ESTABLISHED

An LCSE peer-to-peer logical channel connection has been established. The logical channel may be used to receive incoming data.

C.4.2.5 State transition diagram

The allowed sequence of primitives between the LCSE and the LCSE user is defined here. The allowed sequence of primitives relates to states of the LCSE as viewed from the LCSE user. The allowed sequences are specified separately for each of an outgoing LCSE and an incoming LCSE, as shown in Figure C.10 and Figure C.11 respectively.

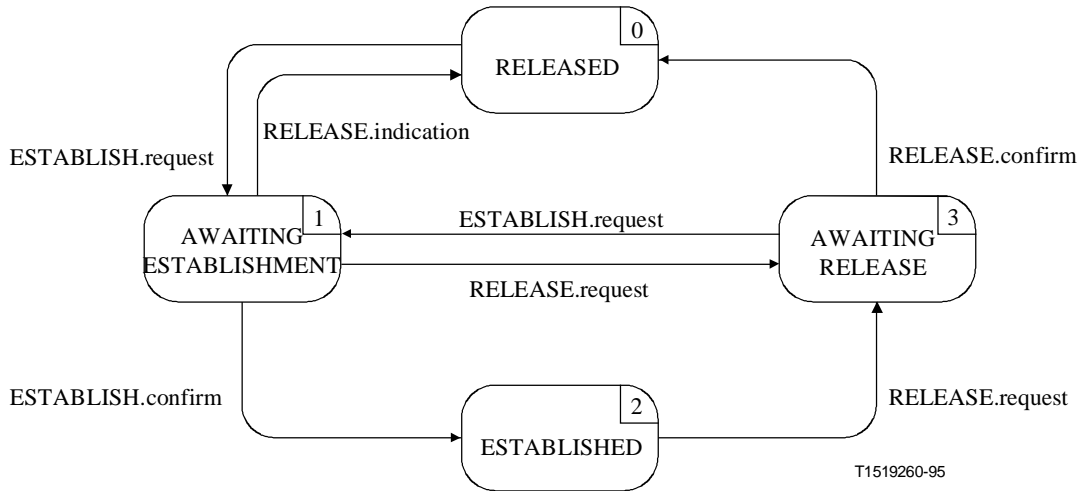


Figure C.10/H.245 – State transition diagram for sequence of primitives at outgoing LCSE

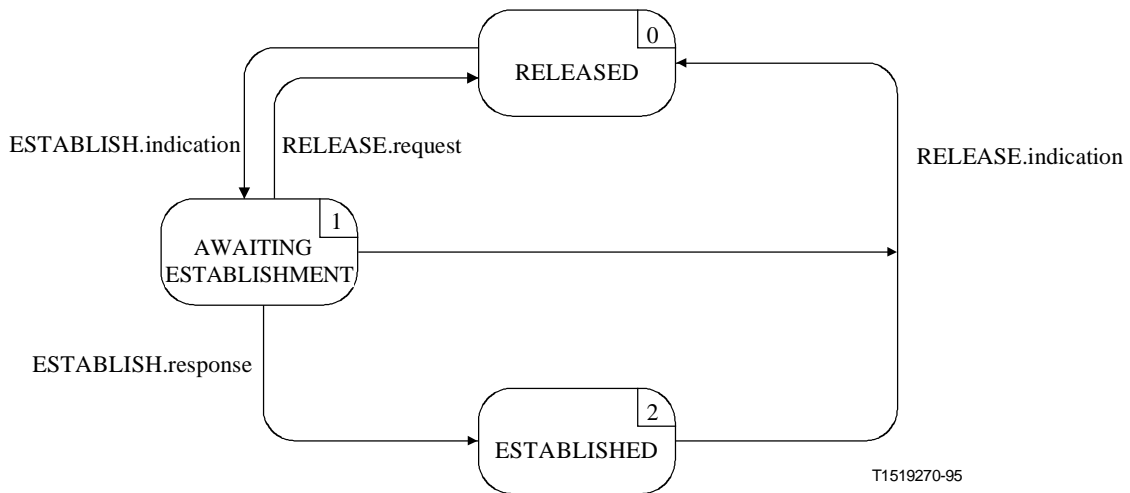


Figure C.11/H.245 – State transition diagram for sequence of primitives at incoming LCSE

C.4.3 Peer-to-peer LCSE communication

C.4.3.1 LCSE messages

Table C.11 shows the LCSE messages and fields, defined in clause 6, which are relevant to the LCSE protocol.

Table C.11/H.245 – LCSE message names and fields

Function	Message	Direction	Field
establishment	OpenLogicalChannel	O → I (Note)	forwardLogicalChannelNumber forwardLogicalChannelParameters
	OpenLogicalChannelAck	O ← I	forwardLogicalChannelNumber
	OpenLogicalChannelReject	O ← I	forwardLogicalChannelNumber cause
release	CloseLogicalChannel	O → I	forwardLogicalChannelNumber source
	CloseLogicalChannelAck	O ← I	forwardLogicalChannelNumber
NOTE – Direction: O – outgoing, I – incoming.			

C.4.3.2 LCSE state variables

The following state variable is defined at the outgoing LCSE:

out_LCN

This state variable distinguishes between outgoing LCSEs. It is initialized at outgoing LCSE initialization. The value of out_LCN is used to set the forwardLogicalChannelNumber field of LCSE messages sent from an outgoing LCSE. For LCSE messages received at an outgoing LCSE, the message forwardLogicalChannelNumber field value is identical to the value of out_LCN.

The following state variable is defined at the incoming LCSE:

in_LCN

This state variable distinguishes between incoming LCSEs. It is initialized at incoming LCSE initialization. The value of in_LCN is used to set the forwardLogicalChannelNumber field of LCSE messages sent from an incoming LCSE. For LCSE messages received at an incoming LCSE, the message forwardLogicalChannelNumber field value is identical to the value of in_LCN.

C.4.3.3 LCSE timers

The following timer is specified for the outgoing LCSE:

T103

This timer is used during the AWAITING ESTABLISHMENT and AWAITING RELEASE states. It specifies the maximum allowed time during which no OpenLogicalChannelAck, OpenLogicalChannelReject or CloseLogicalChannelAck message may be received.

C.4.4 LCSE procedures

C.4.4.1 Introduction

Figure C.12 summarizes the primitives and their parameters, and the messages, for each of the outgoing and incoming LCSE.

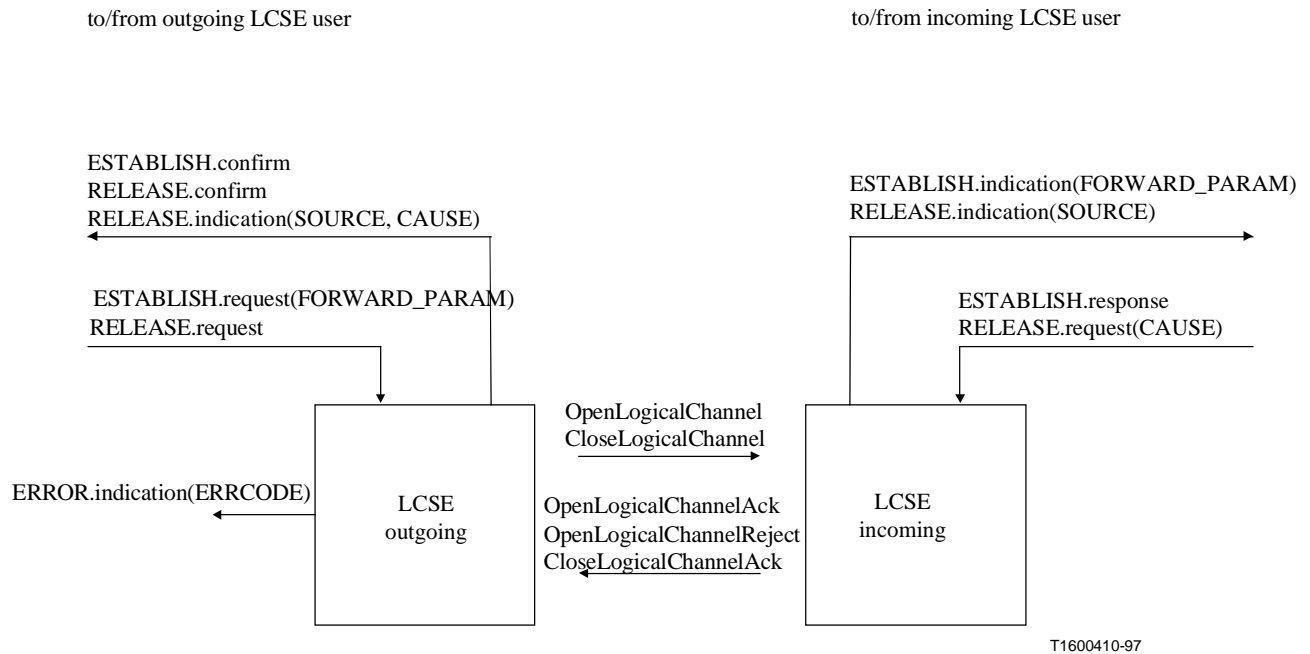


Figure C.12/H.245 – Primitives and messages in the Logical Channel Signalling Entity

C.4.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.12.

Table C.12/H.245 – Default primitive parameter values

Primitive	Parameter	Default value (Note)
ESTABLISH.indication	FORWARD_PARAM	OpenLogicalChannel.forwardLogicalChannelParameters
RELEASE.indication	SOURCE CAUSE	CloseLogicalChannel.source null
NOTE – A primitive parameter shall be coded as null, if an indicated message field is not present in the message.		

C.4.4.3 Message field default values

Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.13.

Table C.13/H.245 – Default message field values

Message	Field	Default value (Note 1)
OpenLogicalChannel (Note 2)	forwardLogicalChannelNumber	out_LCN
	forwardLogicalChannelParameters	ESTABLISH.request(FORWARD_PARAM)
OpenLogicalChannelAck	forwardLogicalChannelNumber	in_LCN
OpenLogicalChannelReject	forwardLogicalChannelNumber cause	in_LCN RELEASE.request(CAUSE)
CloseLogicalChannel	forwardLogicalChannelNumber source	out_LCN user
CloseLogicalChannelAck	forwardLogicalChannelNumber	in_LCN
NOTE 1 – A message field shall not be coded, if the corresponding primitive parameter is null i.e. not present.		
NOTE 2 – reverseLogicalChannelParameters are not coded in unidirectional logical channel signalling procedures.		

C.4.4.4 ERRCODE parameter values

The ERRCODE parameter of the ERROR.indication primitive indicates a particular error condition. Table C.14 shows the values that the ERRCODE parameter may take at the outgoing LCSE. There is no ERROR.indication primitive associated with the incoming LCSE.

Table C.14/H.245 – ERRCODE parameter values at outgoing LCSE

Error type	Error code	Error condition	State
Inappropriate message	A	OpenLogicalChannelAck	RELEASED
	B	OpenLogicalChannelReject	RELEASED ESTABLISHED
	C	CloseLogicalChannelAck	ESTABLISHED
No response from peer LCSE	D	Timer T103 expiry	AWAITING ESTABLISHMENT AWAITING RELEASE

C.4.4.5 SDLs

The outgoing LCSE and the incoming LCSE procedures are expressed in SDL form in Figure C.13 and Figure C.14 respectively.

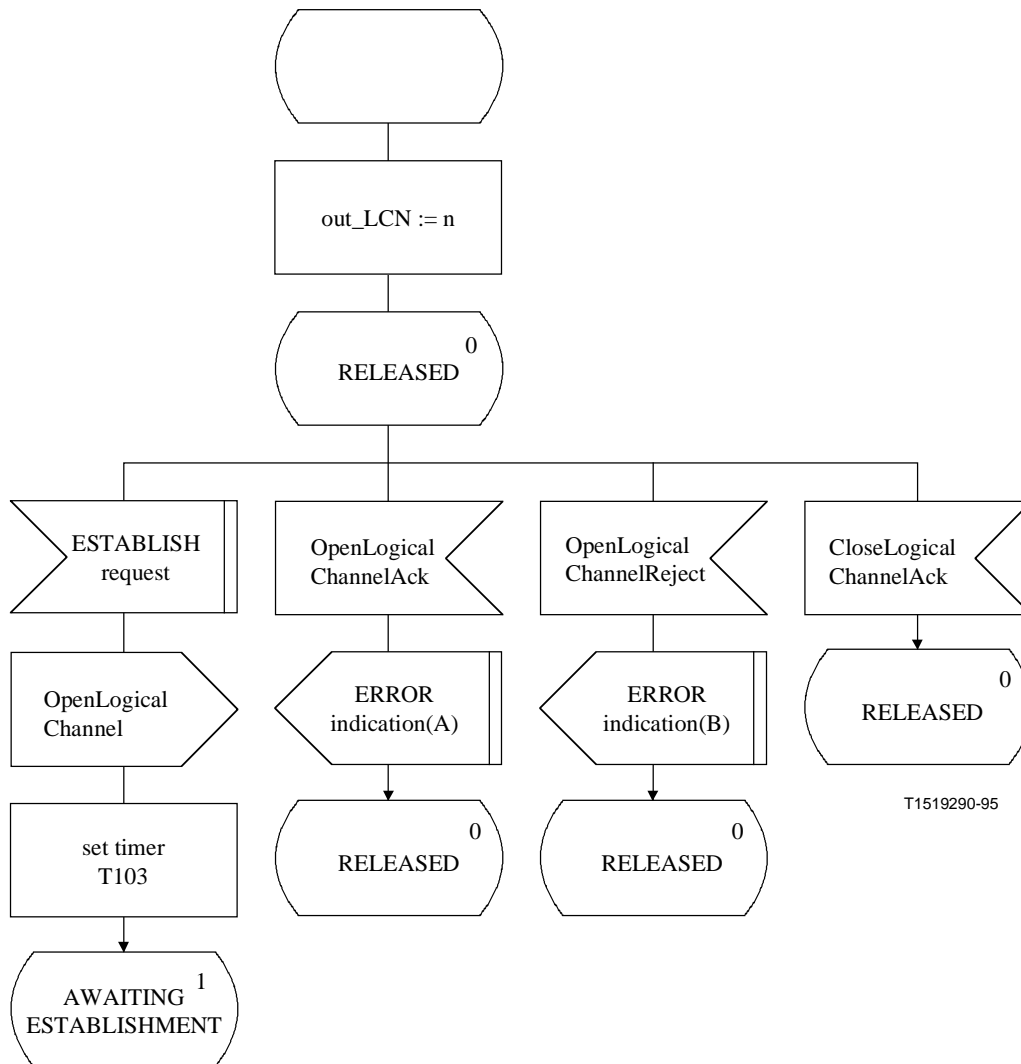
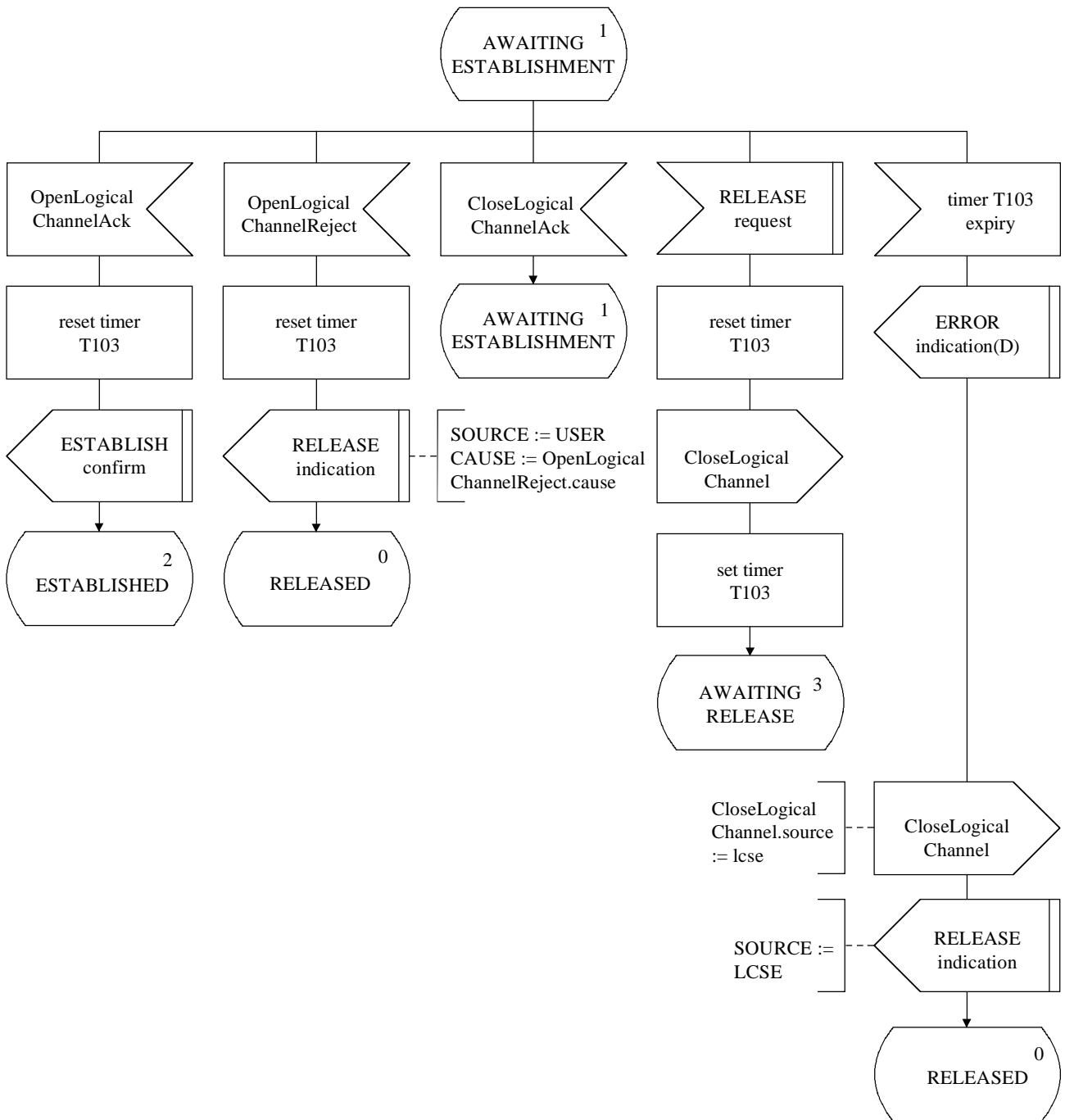
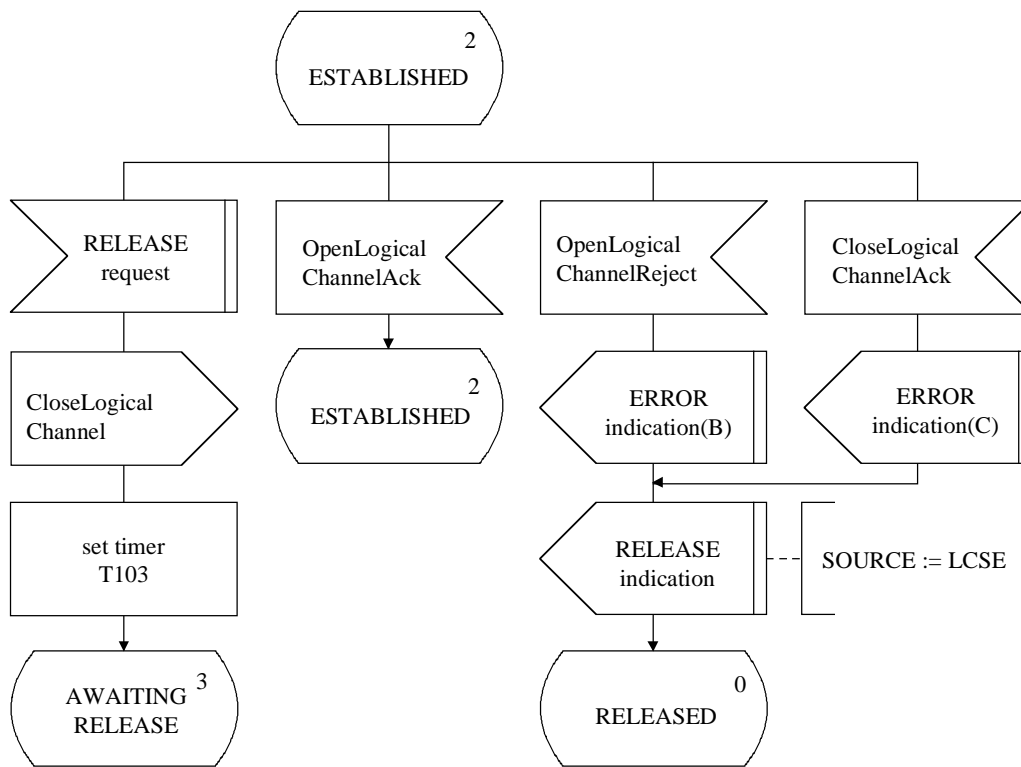


Figure C.13/H.245 – Outgoing LCSE SDL (sheet 1 of 4)



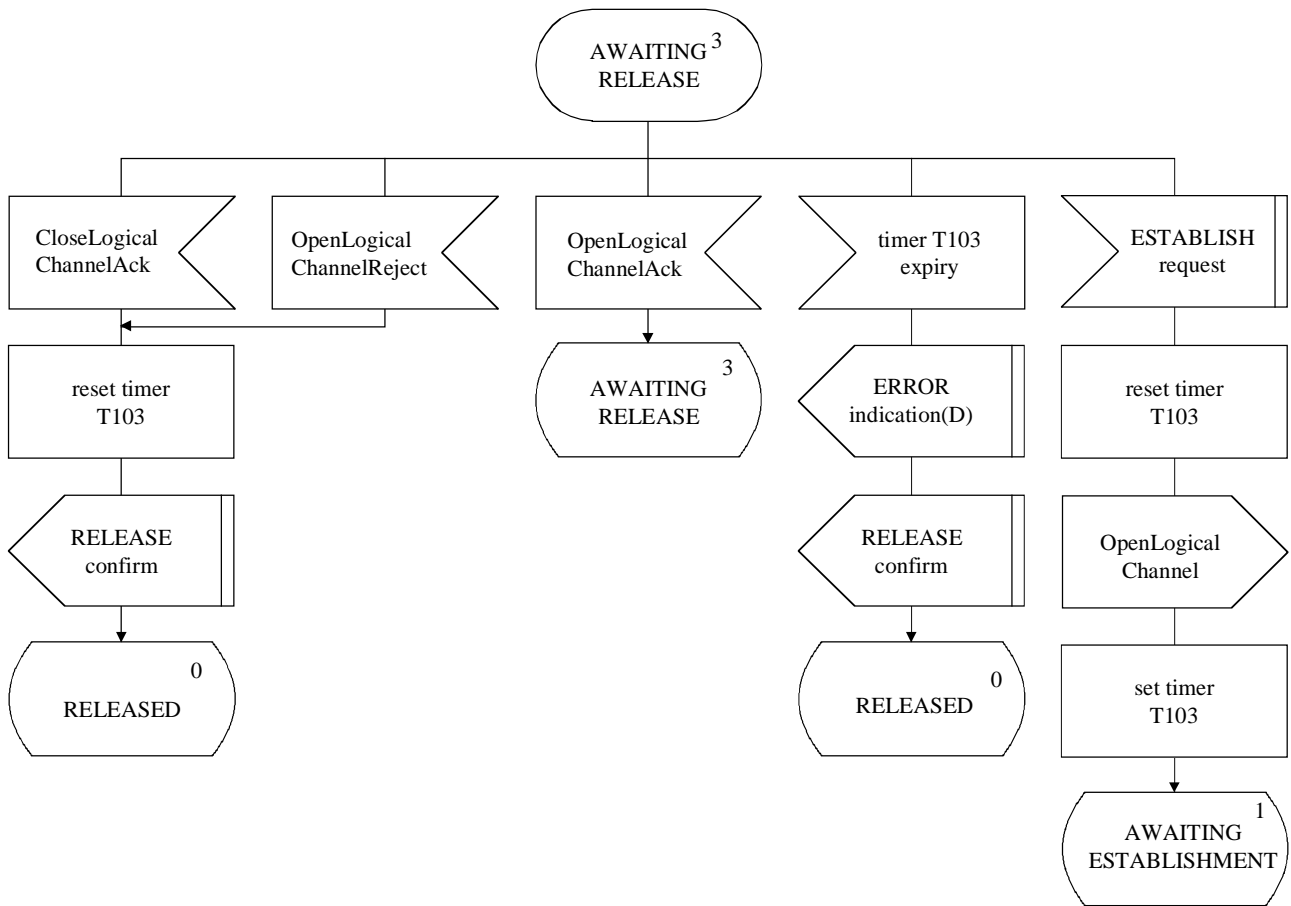
T1600080-97

Figure C.13/H.245 – Outgoing LCSE SDL (sheet 2 of 4)



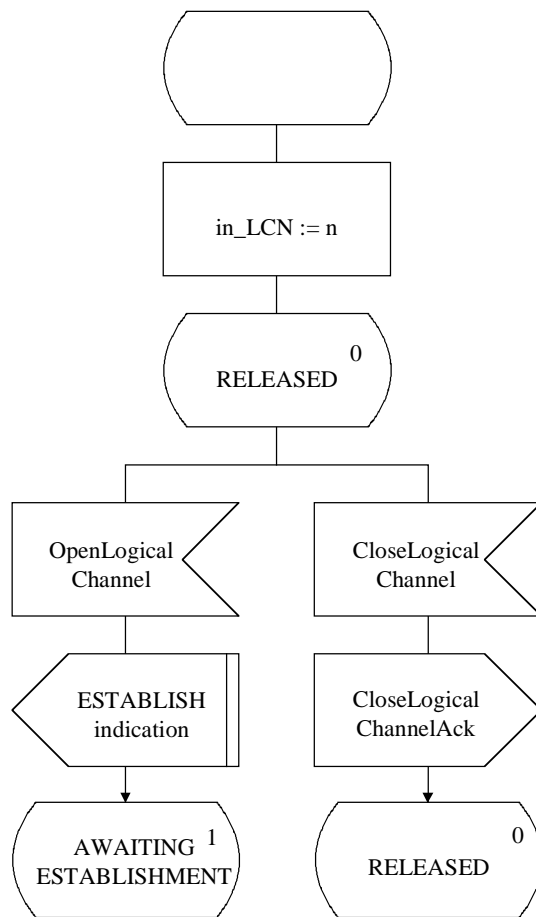
T1519310-95

Figure C.13/H.245 – Outgoing LCSE SDL (sheet 3 of 4)



T1519320-95

Figure C.13/H.245 – Outgoing LCSE SDL (sheet 4 of 4)



T1519330-95

Figure C.14/H.245 – Incoming LCSE SDL (sheet 1 of 3)

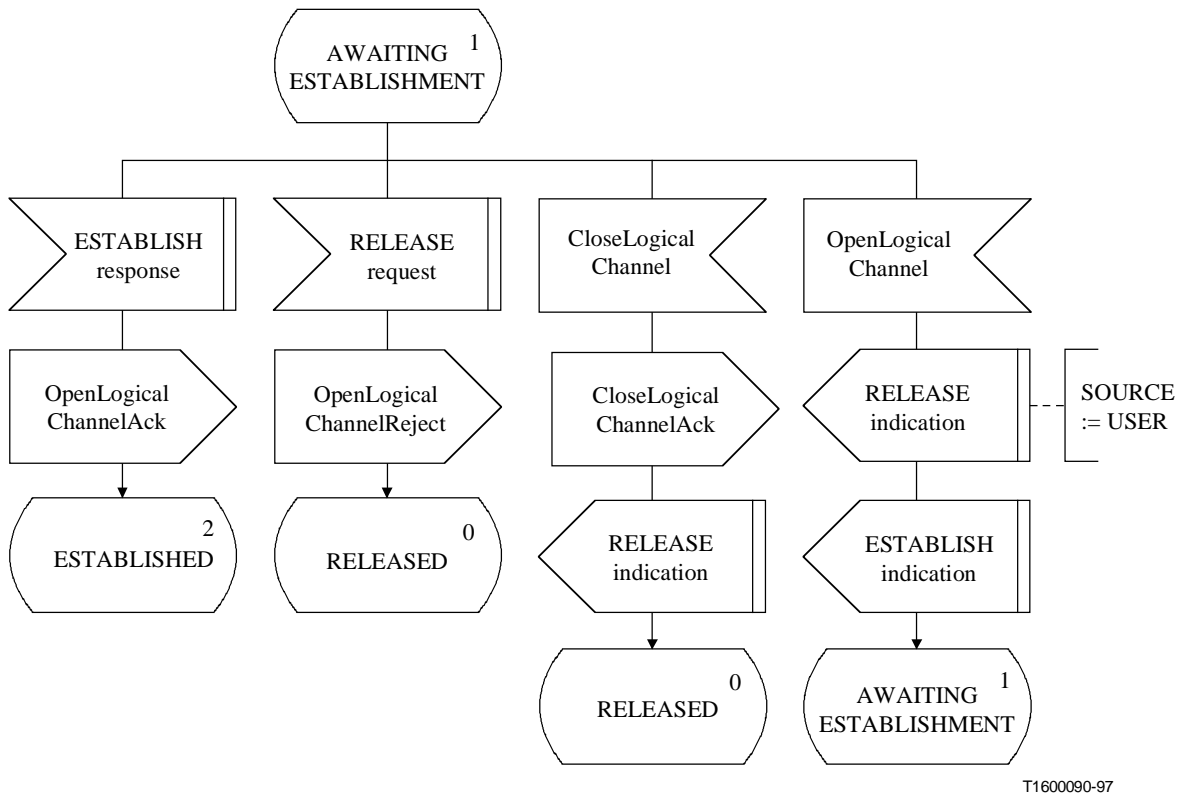


Figure C.14/H.245 – Incoming LCSE SDL (sheet 2 of 3)

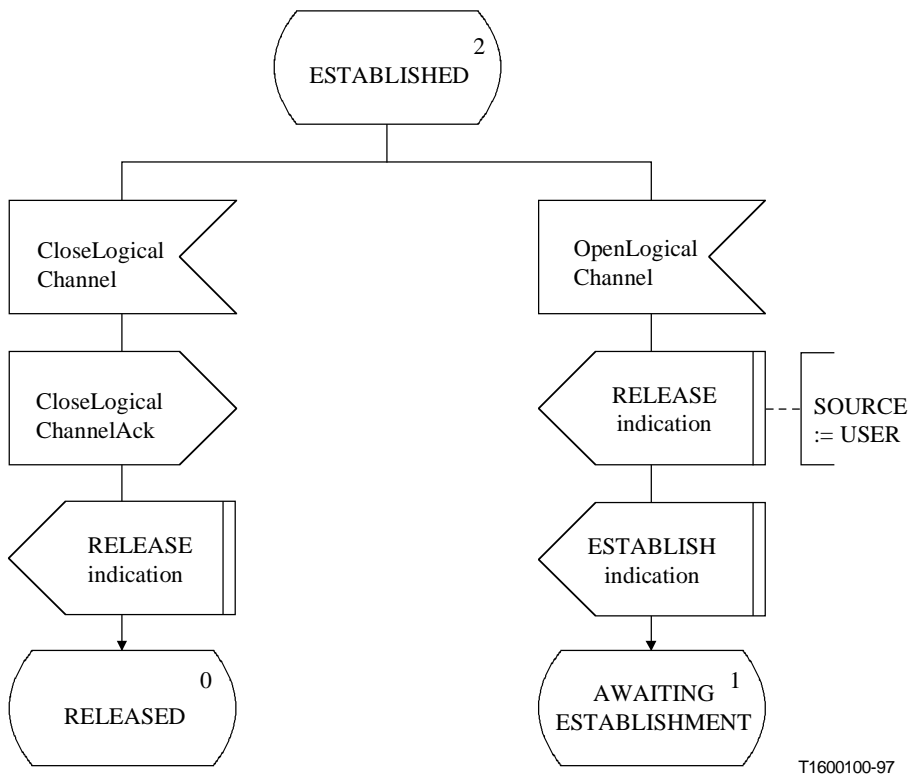


Figure C.14/H.245 – Incoming LCSE SDL (sheet 3 of 3)

C.5 Bidirectional Logical Channel signalling procedures

C.5.1 Introduction

The protocol specified here provides reliable opening and closing of bidirectional logical channels using acknowledged procedures.

The protocol specified here is referred to as the Bidirectional Logical Channel Signalling Entity (B-LCSE). Procedures are specified in terms of primitives at the interface between the B-LCSE and the B-LCSE user, and B-LCSE states. Protocol information is transferred to the peer B-LCSE via relevant messages defined in clause 6.

There is an outgoing B-LCSE and an incoming B-LCSE. At each of the outgoing and incoming sides there is one instance of the B-LCSE for each bidirectional logical channel. There is no connection between an incoming B-LCSE and an outgoing B-LCSE at one side, other than via primitives to and from the B-LCSE user. B-LCSE error conditions are reported.

A bidirectional logical channel consists of a pair of associated unidirectional channels. "Forward" (Outgoing side) is used to refer to transmission in the direction from the terminal making the request for a bidirectional logical channel to the other terminal, and "reverse" (Incoming side) is used to refer to the opposite direction of transmission.

Data shall only be sent on a bidirectional logical channel in the ESTABLISHED state. However, data may be received on the forward channel when the incoming B-LCSE is in the AWAITING CONFIRMATION state. Data that is received while in other states than the ESTABLISHED state and the AWAITING CONFIRMATION state shall be discarded and no fault shall be considered to have occurred.

A terminal may reject a request to open a bidirectional logical channel solely because it cannot support the requested reverse channel parameters. In this case it shall reject the request with cause equal to `unsuitableReverseParameters`, and shall immediately initiate procedures to establish a bidirectional logical channel as requested by the remote terminal, in which the reverse parameters are identical to the forward parameters of the remote terminal's failed request, and with forward parameters that the terminal can support and which the remote terminal is known to be able to support.

Mode switching should be performed by closing and opening existing logical channels, or by opening new logical channels.

NOTE – Some Recommendations that use this Recommendation may define some default logical channels. These shall be considered ESTABLISHED from the start of communication and shall not be opened using these procedures. They may, however, be closed by these procedures, and subsequently be reopened for the same or a different purpose.

A terminal that is no longer capable of processing the signals on a logical channel should take appropriate action: this should include closing the logical channel and transmitting the relevant (changed) capability information to the remote terminal.

The following text provides an overview of the operation of the B-LCSE protocol. In the case of discrepancy between this and the formal specification, the formal specification will supersede.

C.5.1.1 Protocol overview

The opening of a logical channel is initiated when the `ESTABLISH.request` primitive is issued by the user at the outgoing B-LCSE. An `OpenLogicalChannel` message, containing both forward and reverse logical channel parameters, is sent to the peer incoming B-LCSE, and timer T103 is started. If an `OpenLogicalChannelAck` message is received in response to the `OpenLogicalChannel` message then timer T103 is stopped, an `OpenLogicalChannelConfirm` message is sent to the peer incoming

B-LCSE, and the user is informed with the ESTABLISH.confirm primitive that the logical channel has been successfully opened. The logical channel may now be used to transmit and receive user information. If however an OpenLogicalChannelReject message is received in response to the OpenLogicalChannel message then timer T103 is stopped and the user is informed with the RELEASE.indication primitive that the peer B-LCSE user has refused establishment of the logical channel.

If timer T103 expires in this period then the user is informed with the RELEASE.indication primitive, and a CloseLogicalChannel message is sent to the peer incoming B-LCSE.

A logical channel which has been successfully established may be closed when the RELEASE.request primitive is issued by the user at the outgoing B-LCSE. A CloseLogicalChannel message is sent to the peer incoming B-LCSE, and the timer T103 is started. When a CloseLogicalChannelAck message is received, timer T103 is stopped and the user is informed that the logical channel has been successfully closed with the RELEASE.confirm primitive.

If timer T103 expires in this period then the user is informed with the RELEASE.indication primitive.

Before either of the OpenLogicalChannelAck or OpenLogicalChannelReject messages have been received in response to a previously sent OpenLogicalChannel message, the user at the outgoing B-LCSE may close the logical channel using the RELEASE.request primitive.

Before the CloseLogicalChannelAck message is received in response to a previously sent CloseLogicalChannel message, the user at the outgoing B-LCSE may establish a new logical channel by issuing the ESTABLISH.request primitive.

C.5.1.2 Protocol overview – Incoming B-LCSE

When an OpenLogicalChannel message is received at the incoming B-LCSE, the user is informed of the request to open a new logical channel with the ESTABLISH.indication primitive. The incoming B-LCSE user signals acceptance of the request to establish the logical channel by issuing the ESTABLISH.response primitive, and an OpenLogicalChannelAck message is sent to the peer outgoing B-LCSE. The forward channel of the bidirectional logical channel may now be used to receive user information. The incoming B-LCSE user signals rejection of the request to establish the logical channel by issuing the RELEASE.request primitive, and an OpenLogicalChannelReject message is sent to the peer outgoing B-LCSE.

When an OpenLogicalChannelConfirm message is received at the incoming B-LCSE, the user is informed that the bidirectional logical channel is established with the ESTABLISH.confirm primitive. The reverse channel of the bidirectional logical channel may now be used to transmit user information.

A logical channel which has been successfully established may be closed when the CloseLogicalChannel message is received at the incoming B-LCSE. The incoming B-LCSE user is informed with the RELEASE.indication primitive, and the CloseLogicalChannelAck message is sent to the peer outgoing B-LCSE.

C.5.1.3 Conflict resolution

Conflicts may arise when requests to open logical channels are initiated at the same time. It may be possible to determine that there is conflict from knowledge of exchanged capabilities. On other occasions, both terminals may initiate the opening of a bidirectional logical channel for the same purpose, even though the exact parameters requested may be different, and both terminals have sufficient capability for both requests. Terminals shall be capable of detecting when both of these situations have arisen, any shall act as follows.

Before logical channels can be opened, one terminal must be determined as the master terminal, and the other as the slave. The protocol defined in C.2 provides one means to make this decision. The master terminal shall reject immediately any request from the slave that it identifies as a conflicting request. The slave terminal may identify such conflicts, but shall respond to the request from the master terminal, with the knowledge that its earlier request will be rejected.

In the second type of conflict defined above, it is impossible to distinguish when two bidirectional channels are actually wanted from the case when only one is wanted. Terminals shall respond assuming that only one is wanted, but a terminal may subsequently repeat its request if the assumption was incorrect.

C.5.2 Communication between the B-LCSE and the B-LCSE user

C.5.2.1 Primitives between the B-LCSE and the B-LCSE user

Communication between the B-LCSE and the B-LCSE user is performed using the primitives shown in Table C.15.

Table C.15/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
ESTABLISH	FORWARD_PARAM REVERSE_PARAM	FORWARD_PARAM REVERSE_PARAM	REVERSE_DATA	REVERSE_DATA
RELEASE	CAUSE	SOURCE CAUSE	not defined (Note 2)	– (Note 1)
ERROR	not defined	ERRCODE	not defined	not defined
NOTE 1 – "-" means no parameters.				
NOTE 2 – "not defined" means that this primitive does not exist.				

C.5.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The ESTABLISH primitives are used to establish a logical channel for audiovisual and data communication.
- b) The RELEASE primitives are used to release a logical channel.
- c) The ERROR primitive reports B-LCSE errors to a management entity.

C.5.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.15 is as follows:

- a) The FORWARD_PARAM parameter specifies the parameters associated with the forward channel, that is, from the terminal containing the outgoing B-LCSE to the terminal containing the incoming B-LCSE. This parameter is mapped to the forwardLogicalChannelParameters field of the OpenLogicalChannel message and is carried transparently to the peer LCSE user.

- b) The REVERSE_PARAM parameter specifies the parameters associated with the reverse channel, that is, from the terminal containing the incoming B-LCSE to the terminal containing the outgoing B-LCSE. This parameter is mapped to the reverseLogicalChannelParameters field of the OpenLogicalChannel message and is carried transparently to the peer LCSE user.
- c) The REVERSE_DATA parameter specifies some parameters associated with the reverse channel, that is, from the terminal containing the incoming B-LCSE to the terminal containing the outgoing B-LCSE. This parameter is mapped to the reverseLogicalChannelParameters field of the OpenLogicalChannelAck message and is carried transparently to the peer B-LCSE user.
- d) The SOURCE parameter indicates to the B-LCSE user the source of the logical channel release. The SOURCE parameter has the value of "USER" or "B-LCSE", indicating either the B-LCSE user, or the B-LCSE. The latter may occur as the result of a protocol error.
- e) The CAUSE parameter indicates the reason as to why the peer B-LCSE user rejected a request to establish a logical channel. The CAUSE parameter is not present when the SOURCE parameter indicates "B-LCSE".
- f) The ERRCODE parameter indicates the type of B-LCSE error. Table C.19 shows the allowed values of the ERRCODE parameter.

C.5.2.4 B-LCSE states

The following states are used to specify the allowed sequence of primitives between the B-LCSE and the B-LCSE user, and the exchange of messages between peer B-LCSEs. The states are specified separately for each of an outgoing B-LCSE and an incoming B-LCSE. The states for an outgoing B-LCSE are:

State 0: RELEASED

The logical channel is released. The logical channel shall not be used to send or receive data.

State 1: AWAITING ESTABLISHMENT

The outgoing B-LCSE is waiting to establish a logical channel with a peer incoming B-LCSE. The logical channel shall not be used to send or receive data.

State 2: ESTABLISHED

The B-LCSE peer-to-peer logical channel connection has been established. The logical channel may be used to send and receive data.

State 3: AWAITING RELEASE

The outgoing B-LCSE is waiting to release a logical channel with the peer incoming B-LCSE. The logical channel shall not be used to send data, but data may continue to be received.

The states for an incoming B-LCSE are:

State 0: RELEASED

The logical channel is released. The logical channel shall not be used to receive or send data.

State 1: AWAITING ESTABLISHMENT

The incoming B-LCSE is waiting to establish a logical channel with a peer outgoing B-LCSE. The logical channel shall not be used to receive or send data.

State 2: AWAITING CONFIRMATION

The incoming B-LCSE is awaiting confirmation that the logical channel is established with a peer outgoing B-LCSE. The logical channel shall not be used to send data, but data may be received.

State 3: ESTABLISHED

An B-LCSE peer-to-peer logical channel connection has been established. The logical channel may be used to receive and send data.

C.5.2.5 State transition diagram

The allowed sequence of primitives between the B-LCSE and the B-LCSE user is defined here. The allowed sequence of primitives relates to states of the B-LCSE as viewed from the B-LCSE user. The allowed sequences are specified separately for each of an outgoing B-LCSE and an incoming B-LCSE, as shown in Figure C.15 and Figure C.16 respectively.

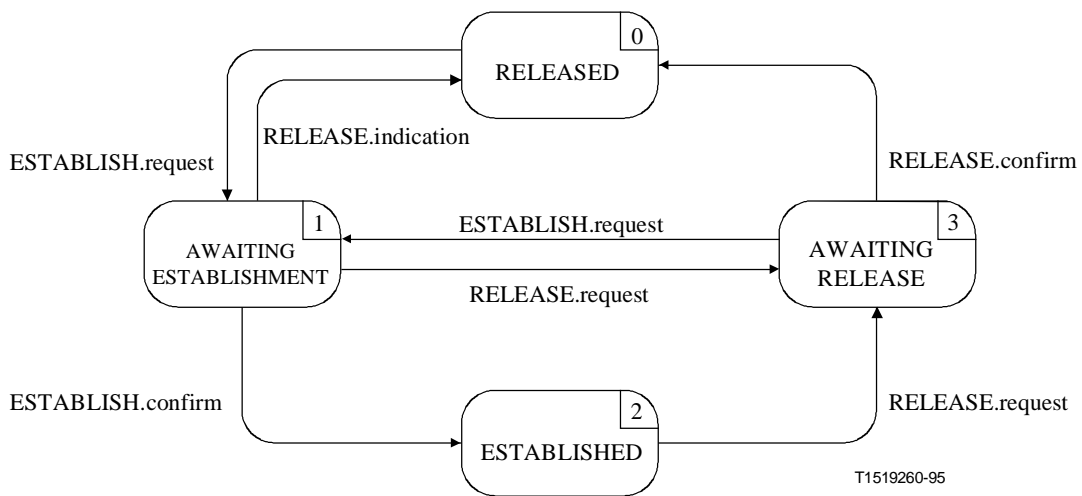


Figure C.15/H.245 – State transition diagram for sequence of primitives at outgoing B-LCSE

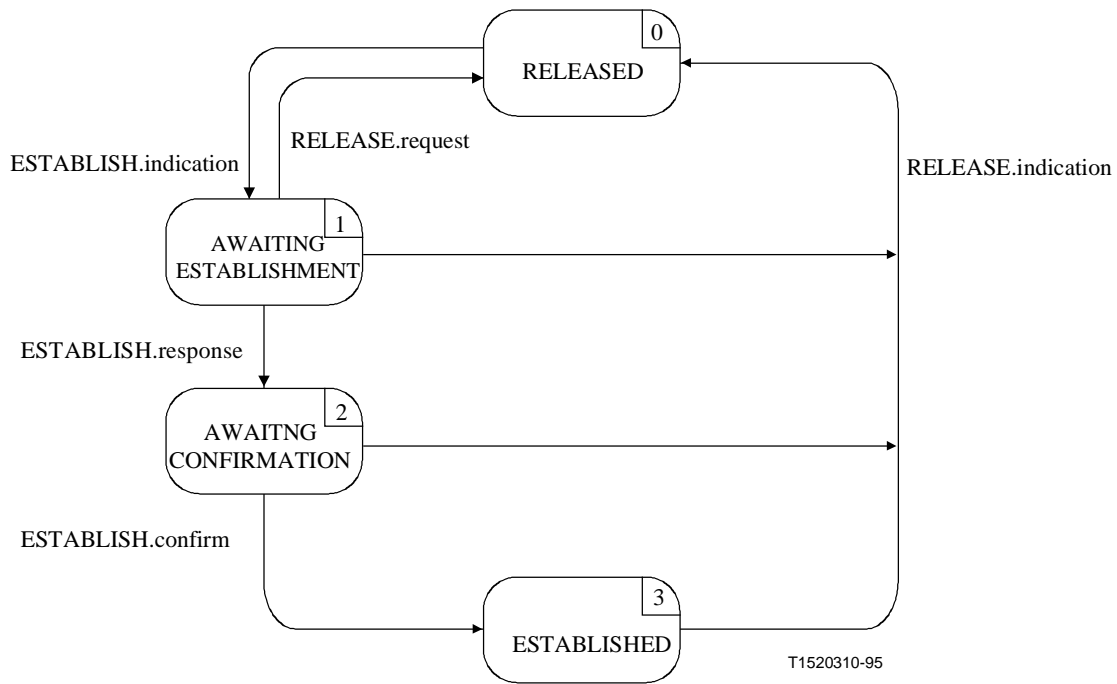


Figure C.16/H.245 –State transition diagram for sequence of primitives at incoming B-LCSE

C.5.3 Peer-to-peer B-LCSE communication

C.5.3.1 B-LCSE messages

Table C.16 shows the B-LCSE messages and fields, defined in clause 6, which are relevant to the B-LCSE protocol.

Table C.16/H.245 – B-LCSE message names and fields

Function	Message	Direction	Field
establishment	OpenLogicalChannel	O → I (Note)	forwardLogicalChannelNumber forwardLogicalChannelParameters reverseLogicalChannelParameters
	OpenLogicalChannelAck	O ← I	forwardLogicalChannelNumber reverseLogicalChannelParameters
	OpenLogicalChannelReject	O ← I	forwardLogicalChannelNumber cause
	OpenLogicalChannelConfirm	O → I	forwardLogicalChannelNumber
release	CloseLogicalChannel	O → I	forwardLogicalChannelNumber
	CloseLogicalChannelAck	O ← I	source forwardLogicalChannelNumber
NOTE – Direction: O – outgoing, I – incoming.			

C.5.3.2 B-LCSE state variables

The following state variable is defined at the outgoing B-LCSE:

out_LCN

This state variable distinguishes between outgoing B-LCSEs. It is initialized at outgoing B-LCSE initialization. The value of out_LCN is used to set the forwardLogicalChannelNumber field of B-LCSE messages sent from an outgoing B-LCSE. For B-LCSE messages received at an outgoing B-LCSE, the message forwardLogicalChannelNumber field value is identical to the value of out_LCN.

The following state variable is defined at the incoming B-LCSE:

in_LCN

This state variable distinguishes between incoming B-LCSEs. It is initialized at incoming B-LCSE initialization. The value of in_LCN is used to set the forwardLogicalChannelNumber field of B-LCSE messages sent from an incoming B-LCSE. For B-LCSE messages received at an incoming B-LCSE, the message forwardLogicalChannelNumber field value is identical to the value of in_LCN.

C.5.3.3 B-LCSE timers

The following timer is specified for the outgoing and incoming B-LCSE:

T103

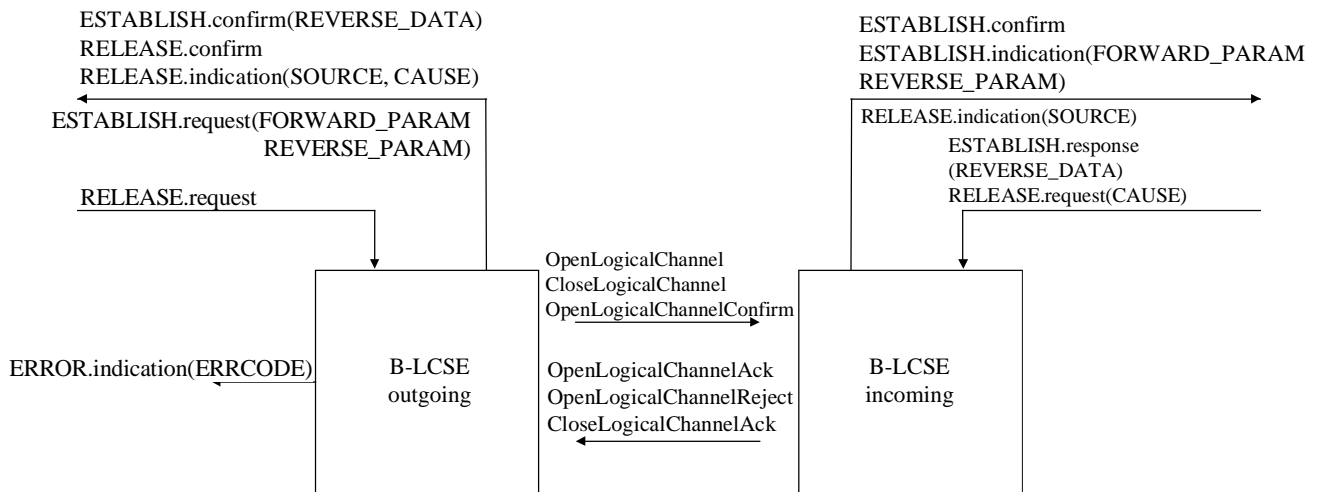
At the outgoing B-LCSE this timer is used during the Awaiting Establishment and Awaiting Release states. It specifies the maximum time during which no OpenLogicalChannelAck, OpenLogicalChannelReject or CloseLogicalChannelAck message may be received.

At the incoming B-LCSE, this timer is used during the Awaiting Confirmation state. It specifies the maximum time during which no OpenLogicalChannelConfirm message may be received.

C.5.4 B-LCSE procedures

C.5.4.1 Introduction

Figure C.17 summarizes the primitives and their parameters, and the messages, for each of the outgoing and incoming B-LCSE.



T1600420-97

Figure C.17/H.245 – Primitives and messages in the Bidirectional Logical Channel Signalling Entity

C.5.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.17.

Table C.17/H.245 – Default primitive parameter values

Primitive	Parameter	Default value (Note)
ESTABLISH.indication	FORWARD_PARAM REVERSE_PARAM	OpenLogicalChannel.forwardLogicalChannelParameters OpenLogicalChannel.reverseLogicalChannelParameters
ESTABLISH.confirm	REVERSE_DATA	OpenLogicalChannelAck.reverseLogicalChannelParameters
RELEASE.indication	SOURCE CAUSE	CloseLogicalChannel.source null
NOTE – A primitive parameter shall be coded as null, if an indicated message field is not present in the message.		

C.5.4.3 Message field default values

Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.18.

Table C.18/H.245 – Default message field values

Message	Field	Default value (Note)
OpenLogicalChannel	forwardLogicalChannelNumber	out_LCN
	forwardLogicalChannelParameters	ESTABLISH.request(FORWARD_PARAM)
	reverseLogicalChannelParameters	ESTABLISH.request(REVERSE_PARAM)
OpenLogicalChannelAck	forwardLogicalChannelNumber	in_LCN
	reverseLogicalChannelParameters	ESTABLISH.response(REVERSE_DATA)
OpenLogicalChannelReject	forwardLogicalChannelNumber	in_LCN
	cause	RELEASE.request(CAUSE)
OpenLogicalChannelConfirm	forwardLogicalChannelNumber	out_LCN
CloseLogicalChannel	forwardLogicalChannelNumber	out_LCN
	source	user
CloseLogicalChannelAck	forwardLogicalChannelNumber	in_LCN

NOTE – A message field shall not be coded, if the corresponding primitive parameter is null i.e. not present.

C.5.4.4 ERRCODE parameter values

The ERRCODE parameter of the ERROR.indication primitive indicates a particular error condition. Table C.19 shows the values that the ERRCODE parameter may take at the outgoing B-LCSE and Table C.20 shows the values that the ERRCODE parameter may take at the incoming B-LCSE.

Table C.19/H.245 – ERRCODE parameter values at outgoing B-LCSE

Error type	Error code	Error condition	State
Inappropriate message	A	OpenLogicalChannelAck	RELEASED
	B	OpenLogicalChannelReject	RELEASED ESTABLISHED
	C	CloseLogicalChannelAck	ESTABLISHED
No response from peer B-LCSE	D	Timer T103 expiry	AWAITING ESTABLISHMENT AWAITING RELEASE

Table C.20/H.245 – ERRCODE parameter values at incoming B-LCSE

Error type	Error code	Error condition	State
Inappropriate message	E	OpenLogicalChannelConfirm	AWAITING ESTABLISHMENT
No response from peer B-LCSE	F	timer T103 expiry	AWAITING CONFIRMATION

C.5.4.5 SDLs

The outgoing B-LCSE and the incoming B-LCSE procedures are expressed in SDL form in Figure C.18 and Figure C.19 respectively.

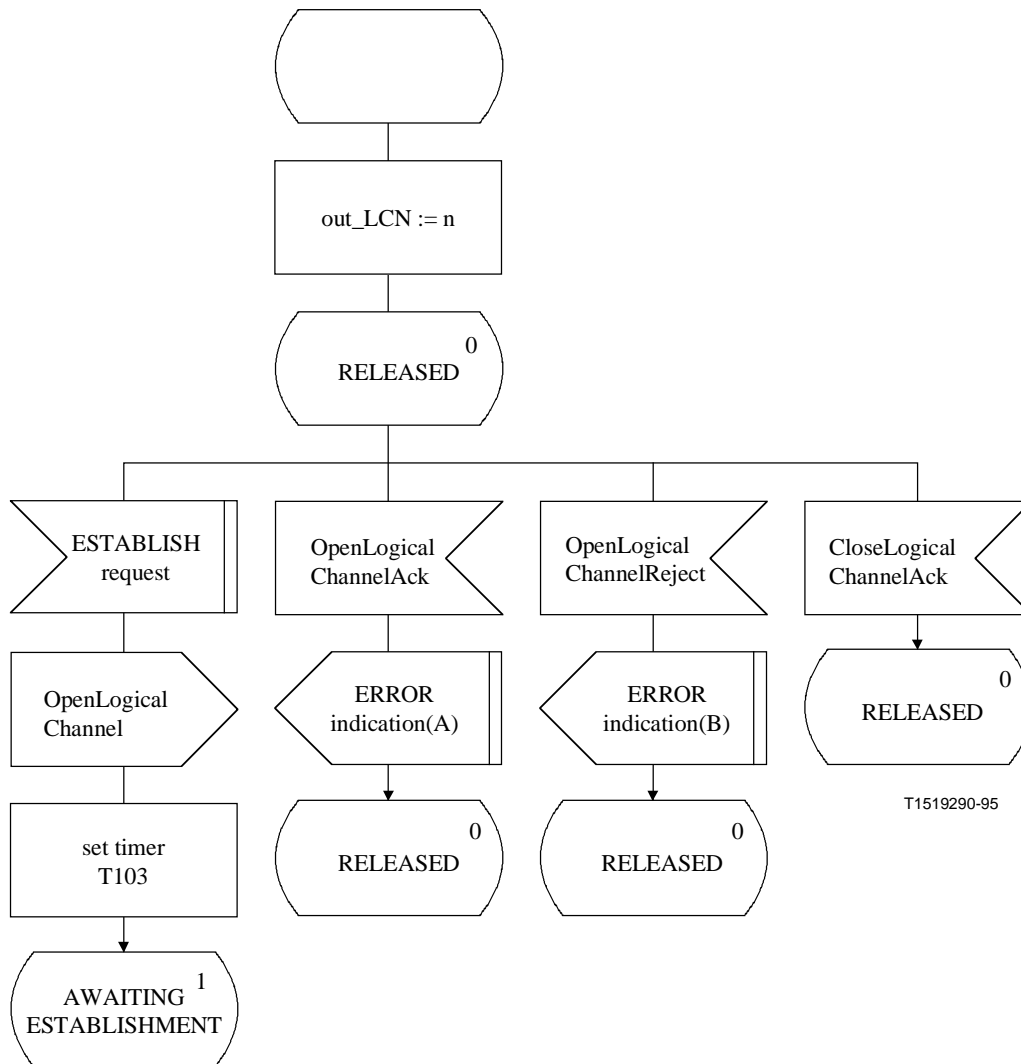
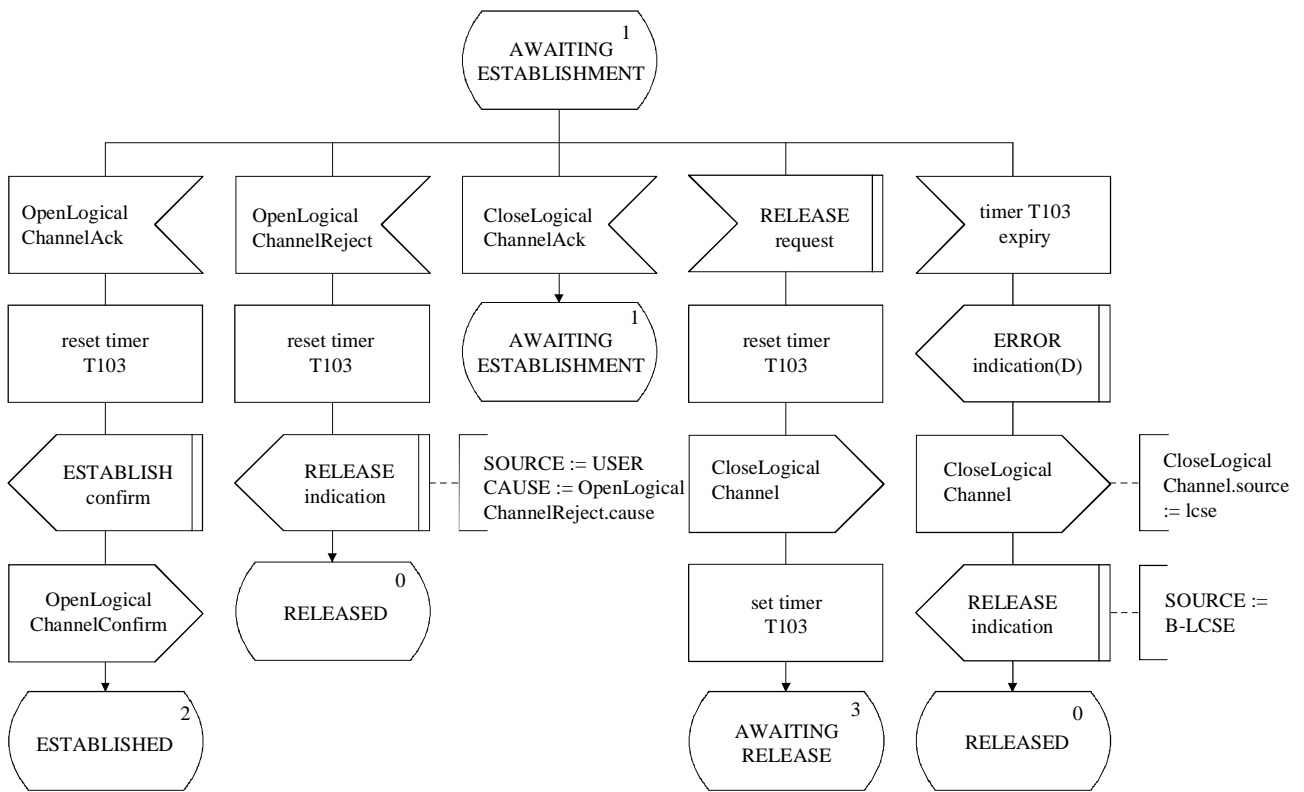
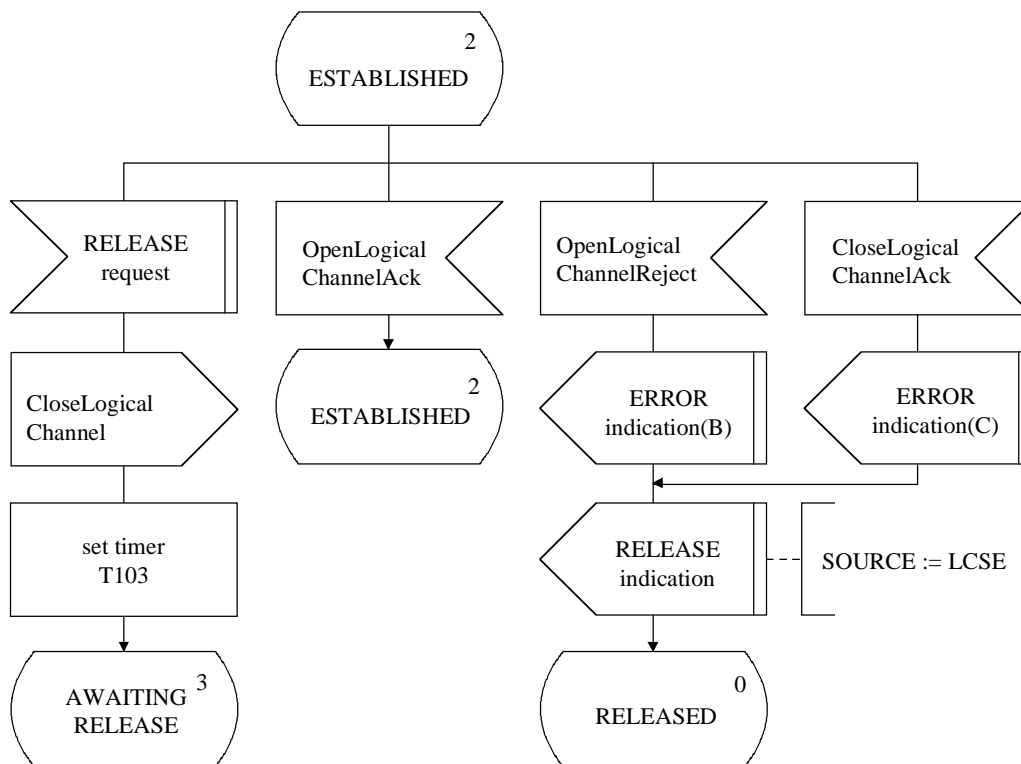


Figure C.18/H.245 – Outgoing B-LCSE SDL (sheet 1 of 4)



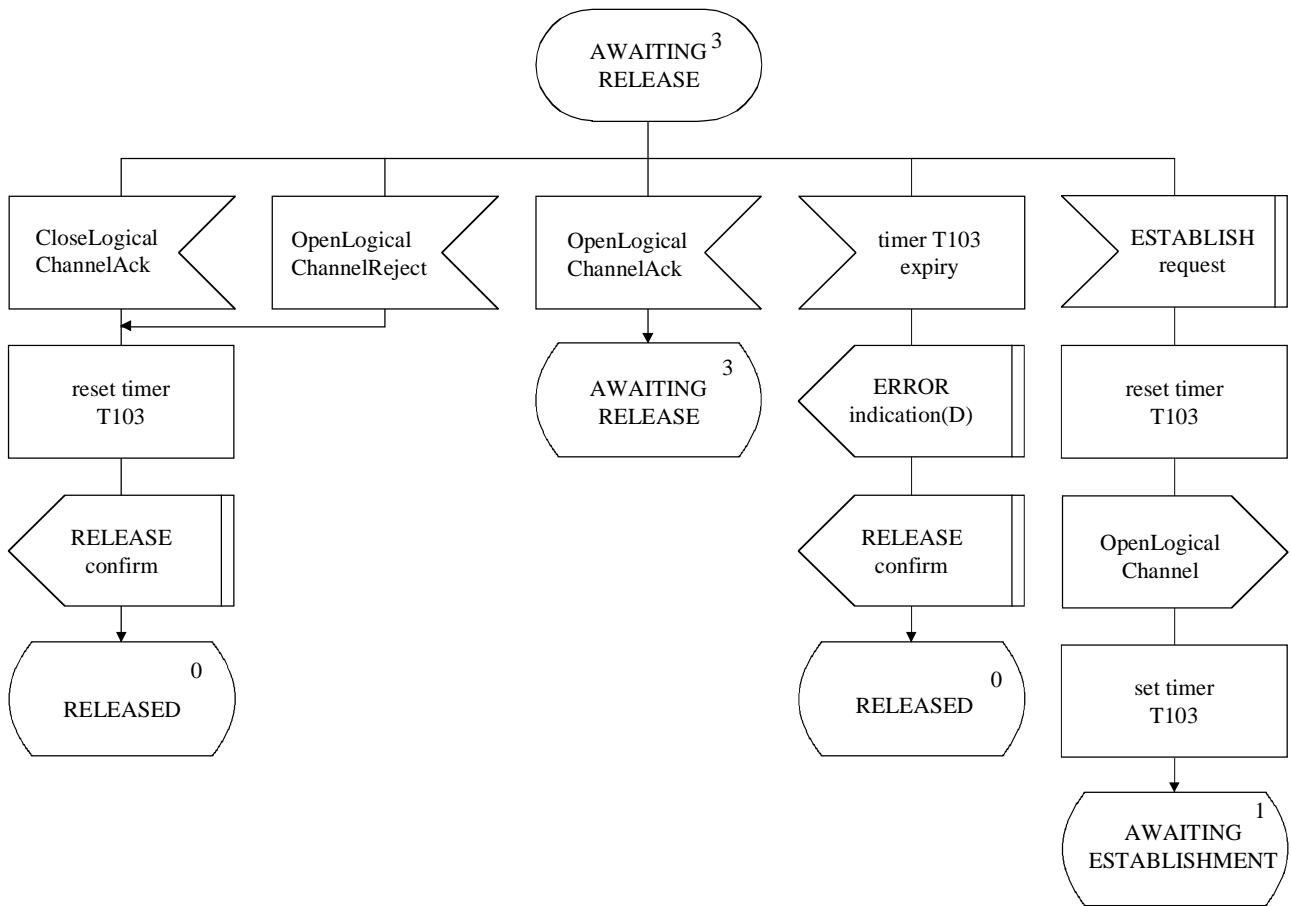
T1600110-97

Figure C.18/H.245 – Outgoing B-LCSE SDL (sheet 2 of 4)



T1519310-95

Figure C.18/H.245 – Outgoing B-LCSE SDL (sheet 3 of 4)



T1519320-95

Figure C.18/H.245 – Outgoing B-LCSE SDL (sheet 4 of 4)

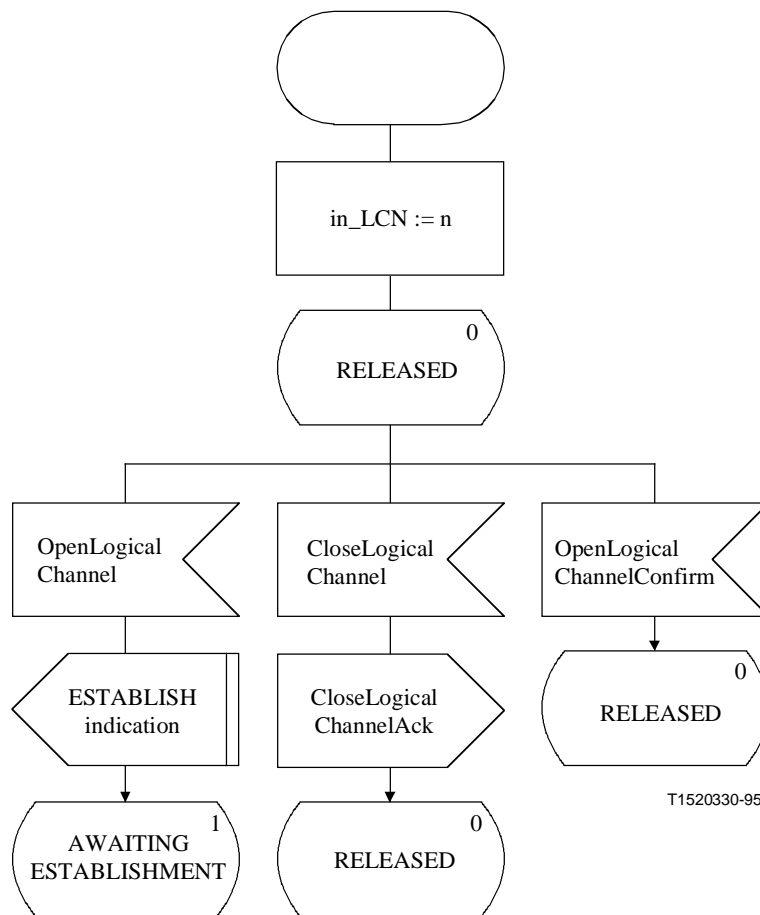


Figure C.19/H.245 – Incoming B-LCSE SDL (sheet 1 of 4)

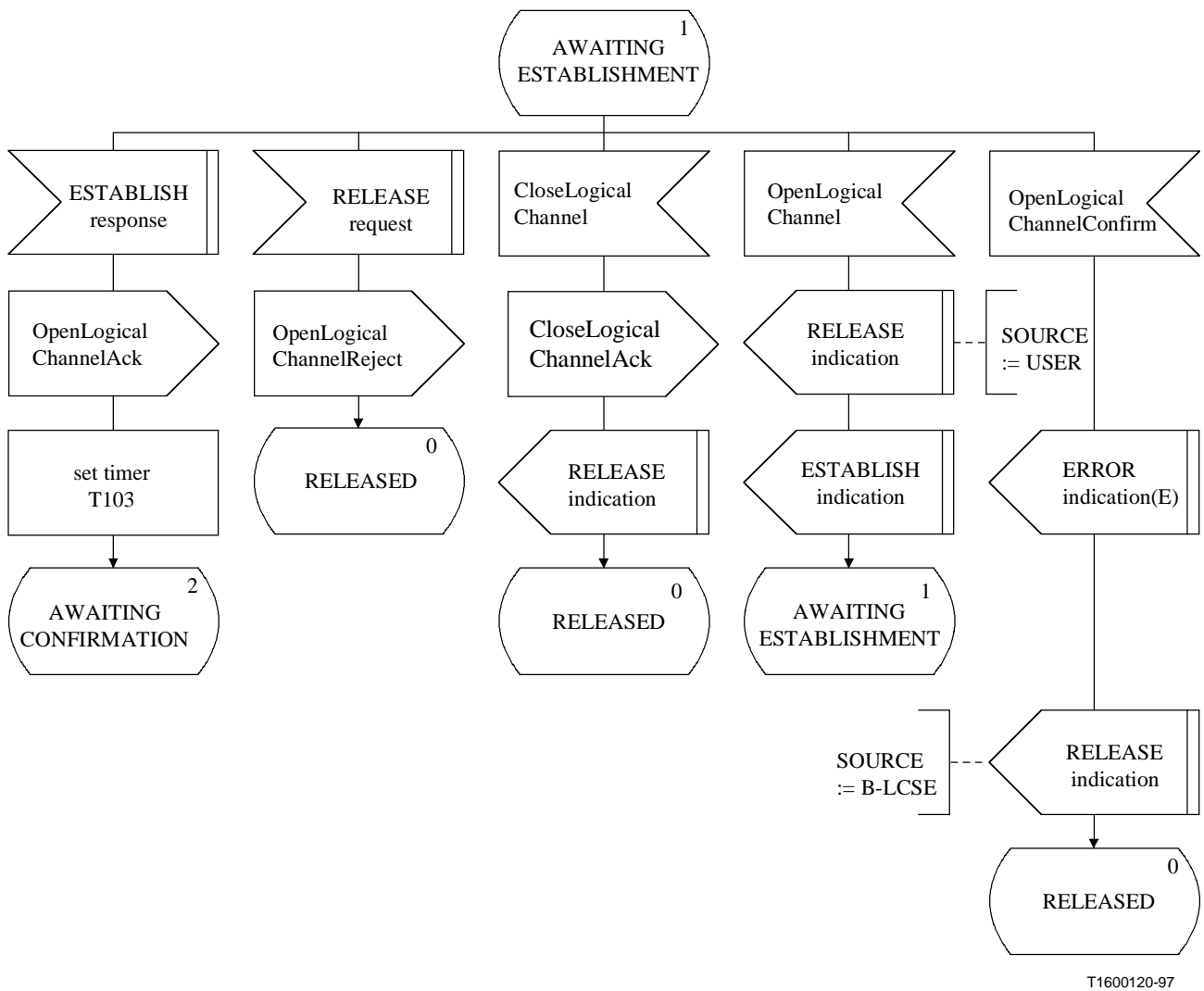


Figure C.19/H.245 – Incoming B-LCSE SDL (sheet 2 of 4)

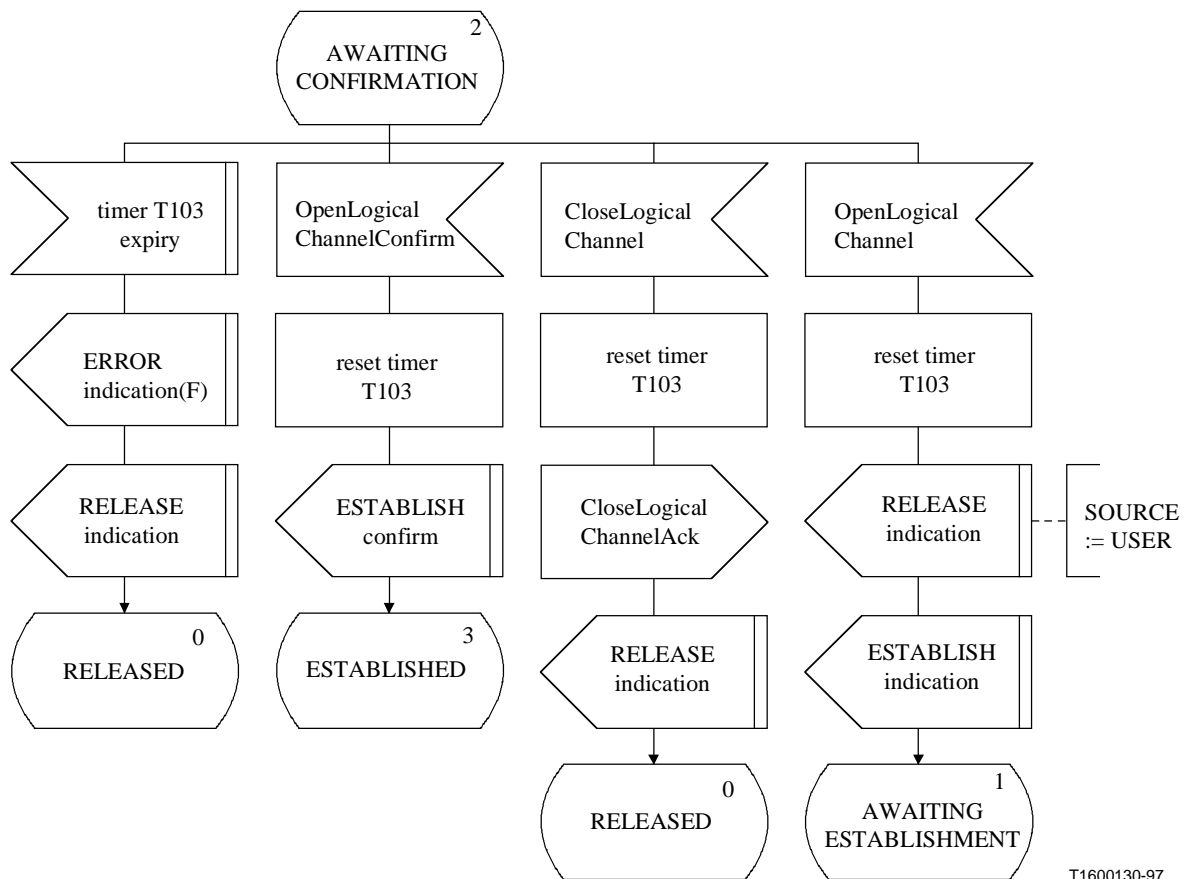


Figure C.19/H.245 – Incoming B-LCSE SDL (sheet 3 of 4)

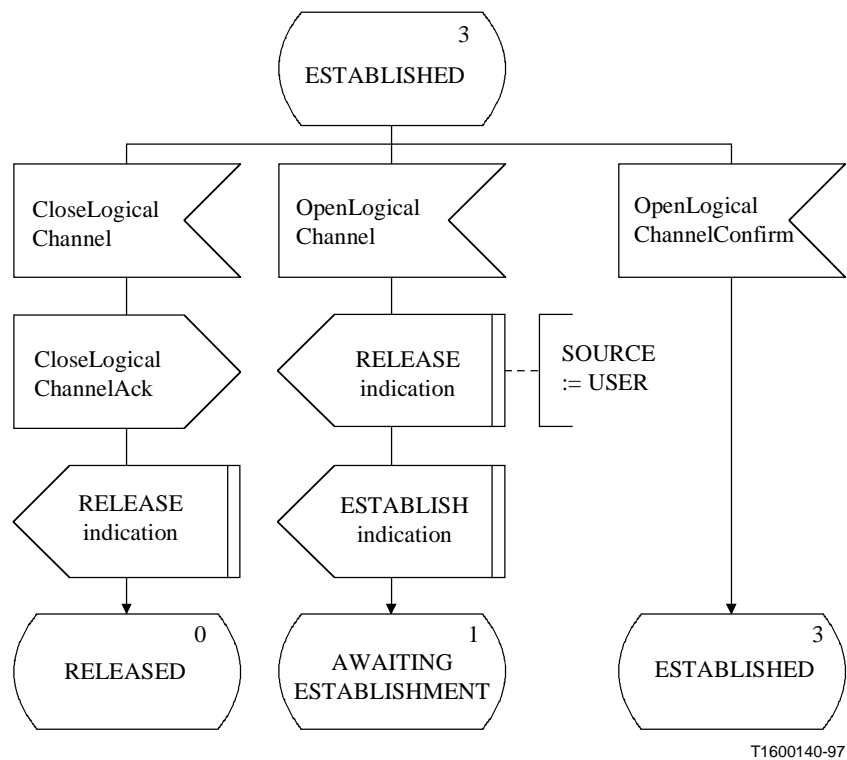


Figure C.19/H.245 – Incoming B-LCSE SDL (sheet 4 of 4)

C.6 Close Logical Channel procedures

C.6.1 Introduction

These procedures are used by a terminal to request the remote terminal to close a logical channel. Note that these are only close request procedures; the actual logical channel close occurs using the LCSE and B-LCSE procedures. The procedures are referred to here as the Close Logical Channel Signalling Entity (CLCSE). Procedures are specified in terms of primitives and states at the interface between the CLCSE and the CLCSE user. Protocol information is transferred to the peer CLCSE via relevant messages defined in clause 6. There is an outgoing CLCSE and an incoming CLCSE. At each of the outgoing and incoming ends there is one instance of the CLCSE for each logical channel.

If a terminal is incapable of processing the incoming signals, it may use these procedures to request the closing of the relevant logical channels.

A terminal that answers such a response positively, that is, by issuing the CLOSE.response primitive, shall initiate the closing of the logical channel by sending the RELEASE.request primitive to the appropriate LCSE or B-LCSE as soon as possible.

The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

C.6.1.1 Protocol overview – Outgoing CLCSE

A close logical channel request procedure is initiated when the CLOSE.request primitive is issued by the user at the outgoing CLCSE. A RequestChannelClose message is sent to the peer incoming CLCSE, and timer T108 is started. If a RequestChannelCloseAck message is received in response to the RequestChannelClose message then timer T108 is stopped and the user is informed with the CLOSE.confirm primitive that the close logical channel request procedure was successful. If however a RequestChannelCloseReject message is received in response to the RequestChannelClose message then timer T108 is stopped and the user is informed with the REJECT.indication primitive that the peer CLCSE user has refused to close the logical channel.

If timer T108 expires then the outgoing CLCSE user is informed with the REJECT.indication primitive and a RequestChannelCloseRelease message is sent.

C.6.1.2 Protocol overview – Incoming CLCSE

When a RequestChannelClose message is received at the incoming CLCSE, the user is informed of the close logical channel request with the CLOSE.indication primitive. The incoming CLCSE user signals acceptance of the close logical channel request by issuing the CLOSE.response primitive, and a RequestChannelCloseAck message is sent to the peer outgoing CLCSE. The incoming CLCSE user signals rejection of the close logical channel request by issuing the REJECT.request primitive, and a RequestChannelCloseReject message is sent to the peer outgoing CLCSE.

C.6.2 Communication between CLCSE and CLCSE user

C.6.2.1 Primitives between CLCSE and CLCSE user

Communication between the CLCSE and CLCSE user, is performed using the primitives shown in Table C.21.

Table C.21/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
CLOSE	– (Note 1)	–	–	–
REJECT	CAUSE	SOURCE CAUSE	not defined (Note 2)	not defined
NOTE 1 – "–" means no parameters.				
NOTE 2 – "not defined" means that this primitive is not defined.				

C.6.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The CLOSE primitives are used to request closure of a logical channel.
- b) The REJECT primitives are used to reject the closing of a logical channel.

C.6.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.21 is as follows:

- a) The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of "USER" or "PROTOCOL". The latter case may occur as the result of a timer expiry.
- b) The CAUSE parameter indicates the reason for refusal to close a logical channel. The CAUSE parameter is not present when the SOURCE parameter indicates "PROTOCOL".

C.6.2.4 CLCSE states

The following states are used to specify the allowed sequence of primitives between the CLCSE and the CLCSE user.

The states for an outgoing CLCSE are:

State 0: IDLE

The CLCSE is idle.

State 1: AWAITING RESPONSE

The CLCSE is waiting for a response from the remote CLCSE.

The states for an incoming CLCSE are:

State 0: IDLE

The CLCSE is idle.

State 1: AWAITING RESPONSE

The CLCSE is waiting for a response from the CLCSE user.

C.6.2.5 State transition diagram

The allowed sequence of primitives between the CLCSE and the CLCSE user is defined here. The allowed sequences are specified separately for each of an outgoing CLCSE and an incoming CLCSE, as shown in Figure C.20 and Figure C.21 respectively.

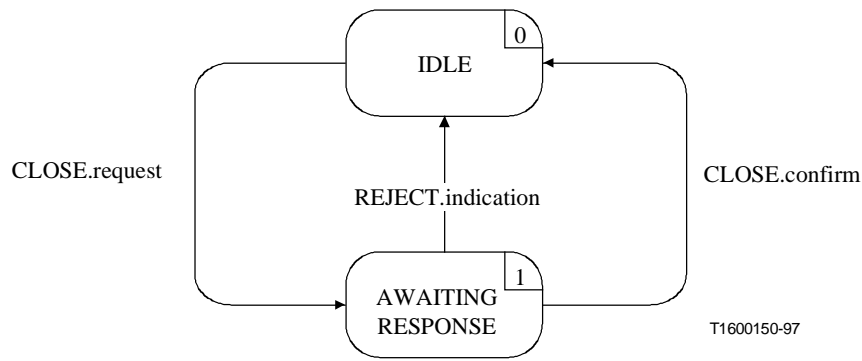


Figure C.20/H.245 – State transition diagram for sequence of primitives at CLCSE outgoing

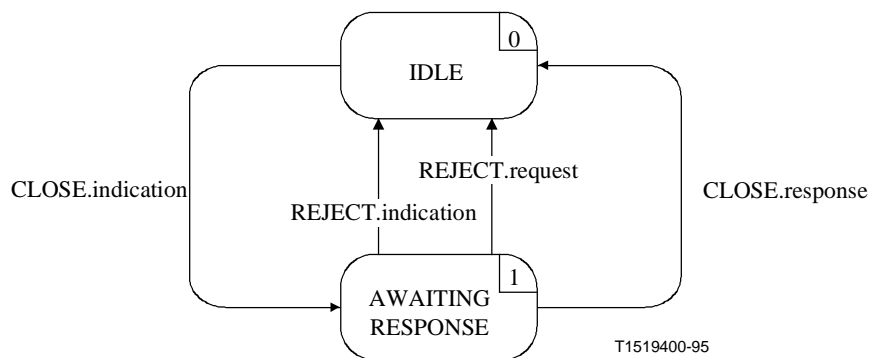


Figure C.21/H.245 – State transition diagram for sequence of primitives at CLCSE incoming

C.6.3 Peer-to-peer CLCSE communication

C.6.3.1 Messages

Table C.22 shows the CLCSE messages and fields, defined in clause 6, which are relevant to the CLCSE protocol.

Table C.22/H.245 – CLCSE message names and fields

Function	Message	Direction	Field
transfer	RequestChannelClose	O → I (Note)	forwardLogicalChannelNumber
	RequestChannelCloseAck	O ← I	forwardLogicalChannelNumber
	RequestChannelCloseReject	O ← I	forwardLogicalChannelNumber
reset	RequestChannelCloseRelease	O → I	forwardLogicalChannelNumber

NOTE – Direction: O – outgoing, I – incoming.

C.6.3.2 CLCSE state variables

The following state variable is defined at the outgoing CLCSE:

out_LCN

This state variable distinguishes between outgoing CLCSEs. It is initialized at outgoing CLCSE initialization. The value of out_LCN is used to set the forwardLogicalChannelNumber field of CLCSE messages sent from an outgoing CLCSE. For CLCSE messages received at an outgoing CLCSE, the message forwardLogicalChannelNumber field value is identical to the value of out_LCN.

The following state variable is defined at the incoming CLCSE:

in_LCN

This state variable distinguishes between incoming CLCSEs. It is initialized at incoming CLCSE initialization. The value of in_LCN is used to set the forwardLogicalChannelNumber field of CLCSE messages sent from an incoming CLCSE. For CLCSE messages received at an incoming CLCSE, the message forwardLogicalChannelNumber field value is identical to the value of in_LCN.

C.6.3.3 CLCSE timers

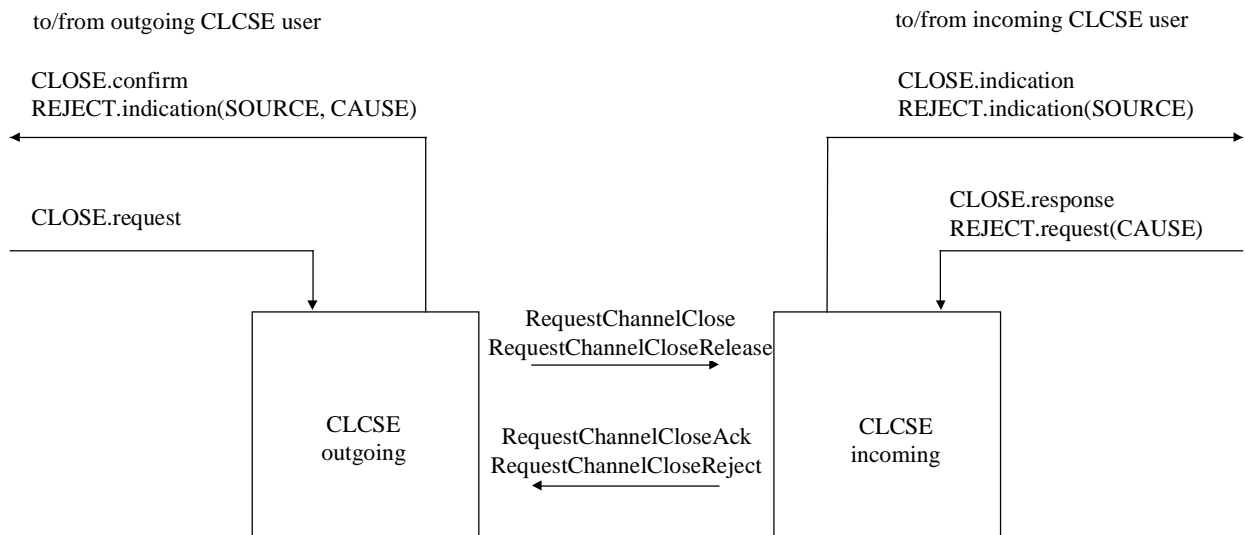
The following timer is specified for the outgoing CLCSE:

T108

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RequestChannelCloseAck or RequestChannelCloseReject message may be received.

C.6.4 CLCSE procedures

Figure C.22 summarizes the CLCSE primitives and their parameters, and messages, for each of the outgoing and incoming CLCSE.



T1519410-95

Figure C.22/H.245 – Primitives and messages in the Close Logical Channel Signalling Entity

C.6.4.1 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.23.

Table C.23/H.245 – Default primitive parameter values

Primitive	Parameter	Default value
REJECT.indication	SOURCE	USER
	CAUSE	null

C.6.4.2 Message field default values

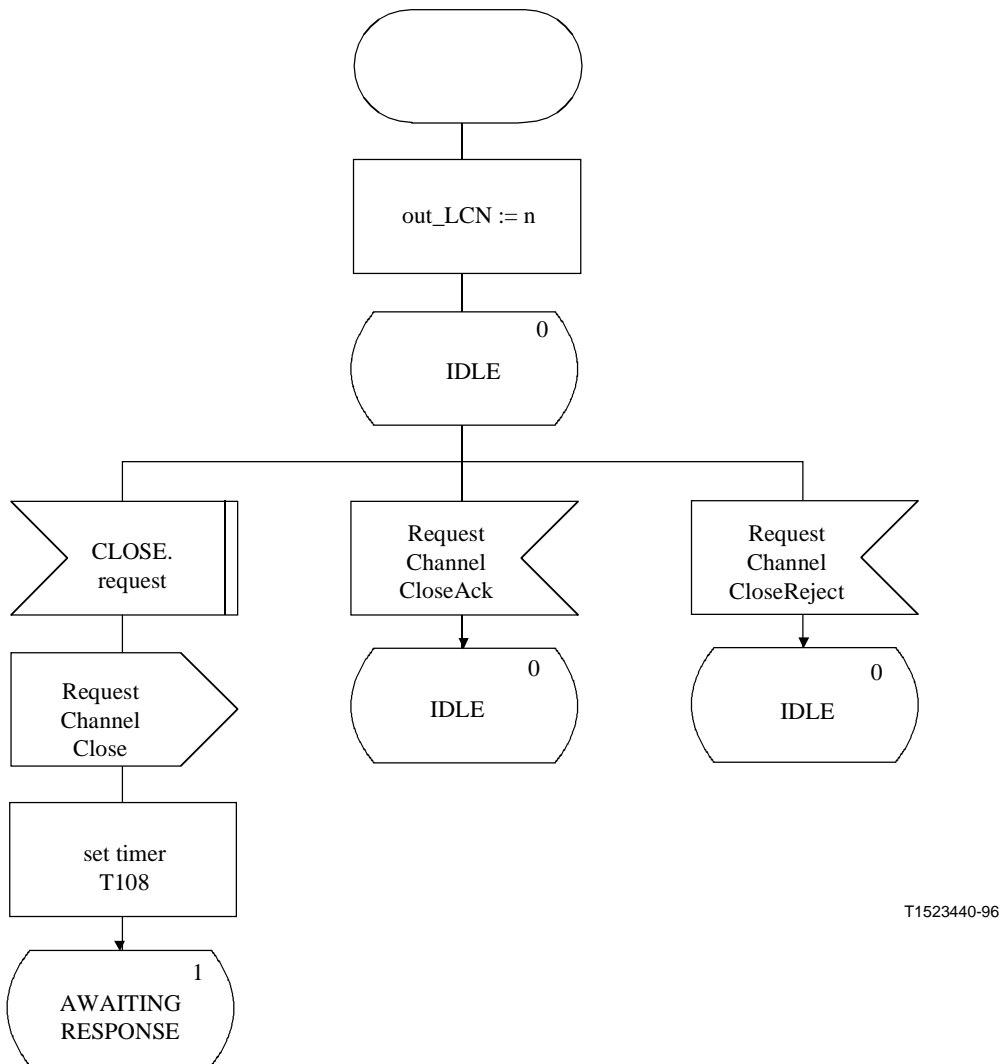
Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.24.

Table C.24/H.245 – Default message field values

Message	Field	Default value
RequestChannelClose	forwardLogicalChannelNumber	out_LCN
RequestChannelCloseAck	forwardLogicalChannelNumber	in_LCN
RequestChannelCloseReject	forwardLogicalChannelNumber cause	in_LCN REJECT.request(CAUSE)
RequestChannelCloseRelease	forwardLogicalChannelNumber	out_LCN

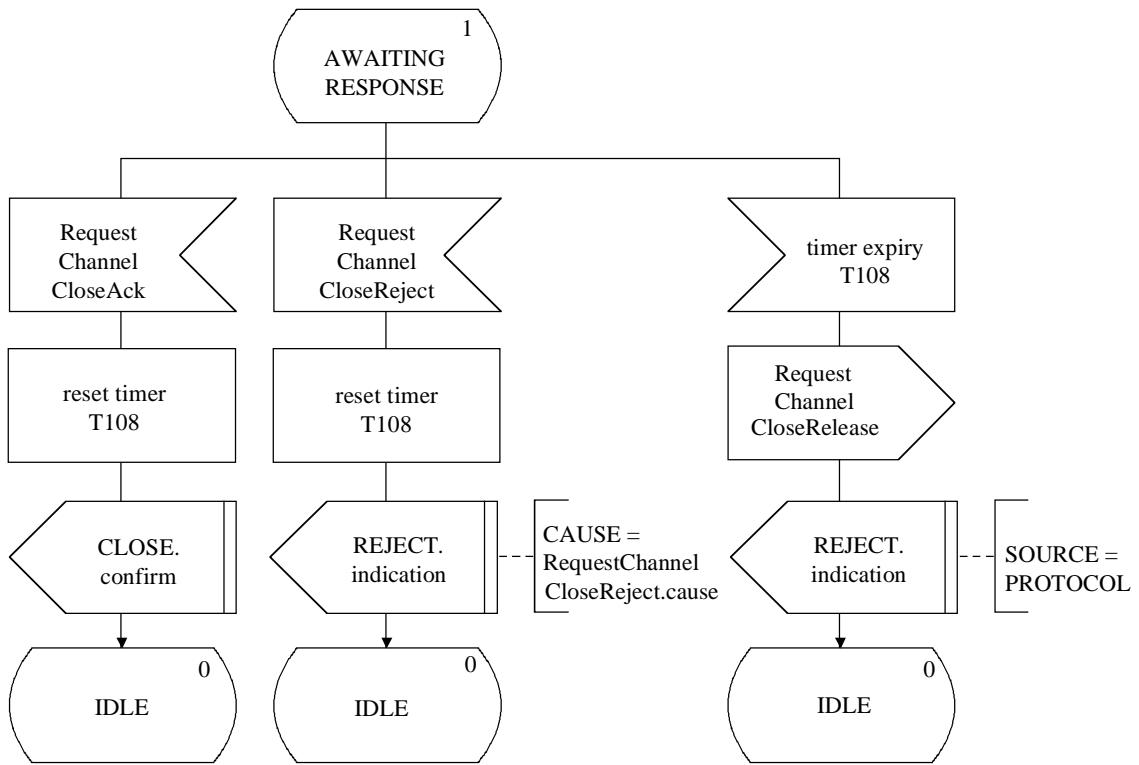
C.6.4.3 SDLs

The outgoing CLCSE and the incoming CLCSE procedures are expressed in SDL form in Figure C.23 and Figure C.24 respectively.



T1523440-96

Figure C.23/H.245 – Outgoing CLCSE SDL (sheet 1 of 2)



T1523450-96

Figure C.23/H.245 – Outgoing CLCSE SDL (sheet 2 of 2)

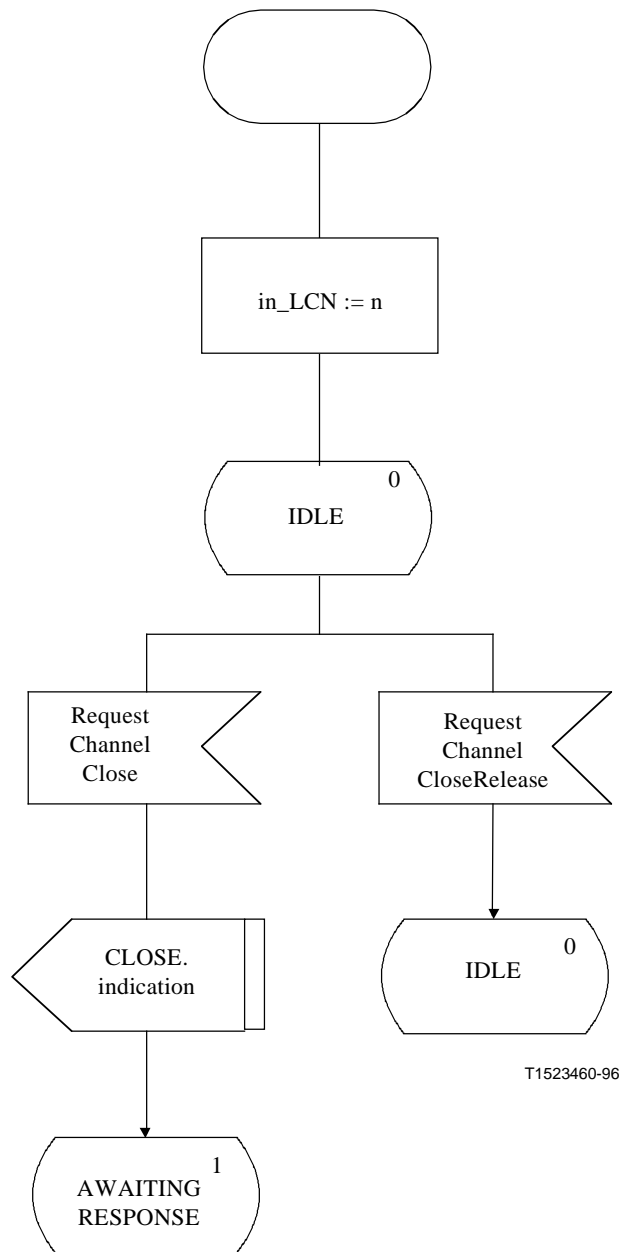
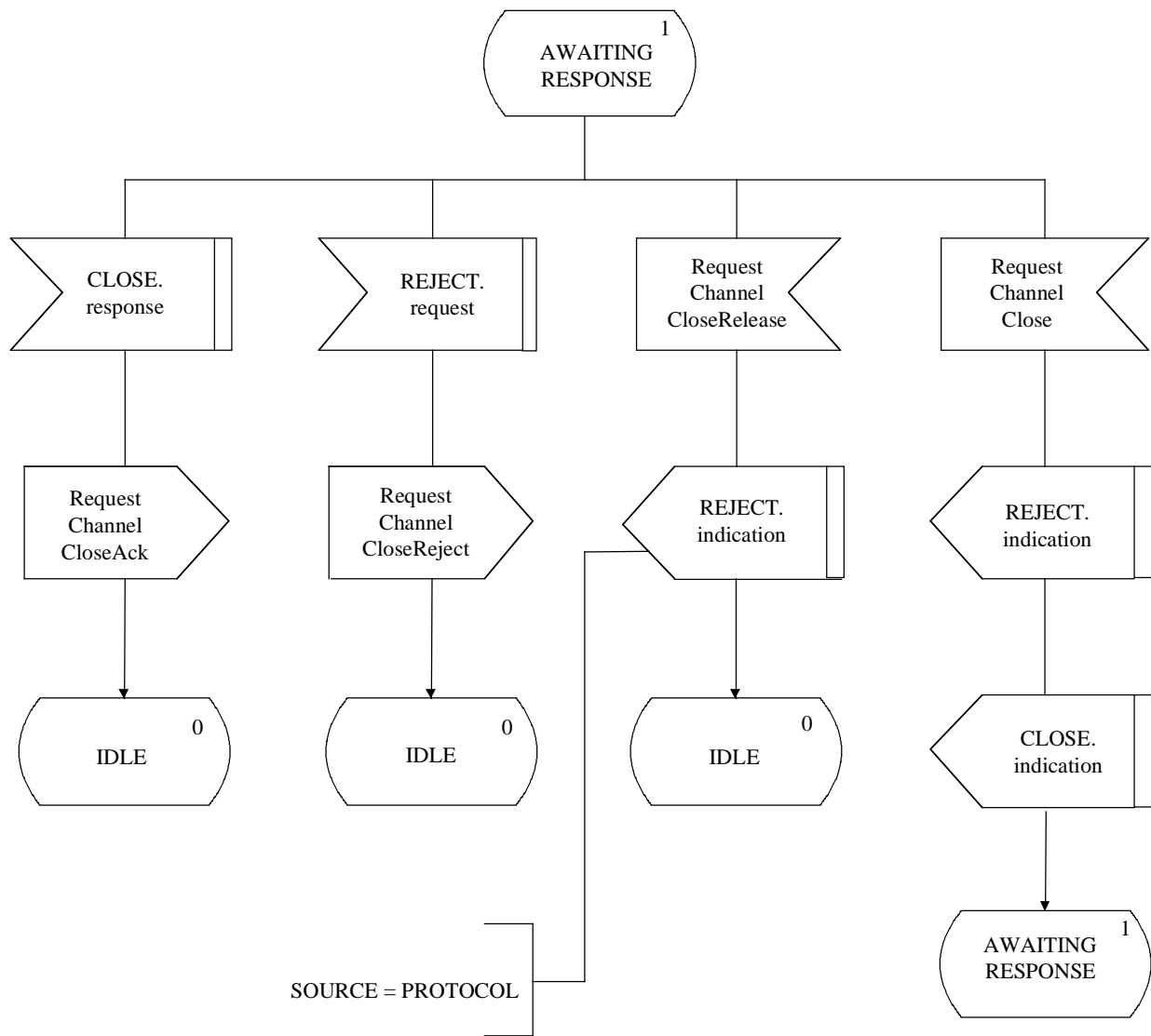


Figure C.24/H.245 – Incoming CLCSE SDL (sheet 1 of 2)



T1523470-96

Figure C.24/H.245 – Incoming CLCSE SDL (sheet 2 of 2)

C.7 H.223 multiplex table procedures

C.7.1 Introduction

The multiplex table serves to associate each octet within a H.223 MUX-PDU [10] with a particular logical channel number. The H.223 multiplex table may have up to 16 entries, numbered from 0 to 15. Table entries 1 to 15 shall be sent from transmitters to receivers as specified in the following procedures.

The procedures described here are referred to as the Multiplex Table Signalling Entity (MTSE). Procedures are specified in terms of primitives and states at the interface between the MTSE and the MTSE user. Protocol information is transferred to the peer MTSE via relevant messages defined in clause 6.

There is an outgoing MTSE and an incoming MTSE. There is one instance of the MTSE for each multiplex table entry.

A transmit terminal uses this protocol to signal to a remote terminal one or more new multiplex table entries. The remote terminal may accept or reject the new multiplex table entries. If the remote

terminal accepts a multiplex table entry, the previous entry at the given entry number is replaced with the new entry.

The transmitter may deactivate a multiplex table entry by sending a MultiplexEntryDescriptor with no elementList. The transmitter shall at no time use a multiplex table entry that is deactivated. Before transmitting a MultiplexEntrySend, the transmitter shall stop using the entries that are described by it. It shall not restart using those entries until it has received a MultiplexEntrySendAck. This procedure is used because if the use of these multiplex table entries is not stopped before sending the MultiplexEntrySend, errors may cause an ambiguity in the receiver.

The transmitter shall stop using deactivated entries before sending the MultiplexEntrySend indicating that they have been deactivated. Deactivated entries may be used again at any time by transmitting a MultiplexEntrySend message for activating that entry. Deactivating entries that are no longer required by the transmitter may increase the probability of detecting errors in the H.223 Multiplex Code field.

NOTE – While some multiplex table entries are being updated, other (active) entries may continue to be used. Also, a multiplex table entry may be deleted in the same MultiplexEntrySend that is used to modify other multiplex table entries.

At the start of communication, unless specified otherwise in an appropriate Recommendation, only table entry 0 is available for transmission, and table entries 1 to 15 are deactivated.

A Request Multiplex Entry procedure may be used at any time to elicit retransmission of specified multiplex table entries from the remote terminal, for example, following an interruption or other cause for uncertainty.

The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

C.7.1.1 Protocol overview – Outgoing MTSE

A multiplex table entry send request procedure is initiated when the TRANSFER.request primitive is issued by the user at the outgoing MTSE. A MultiplexEntrySend message is sent to the peer incoming MTSE, and timer T104 is started. If a MultiplexEntrySendAck message is received in response to the MultiplexEntrySend message then timer T104 is stopped and the user is informed with the TRANSFER.confirm primitive that the multiplex table entry send request was successful. If however a MultiplexEntrySendReject message is received in response to the MultiplexEntrySend message then timer T104 is stopped and the user is informed with the REJECT.indication primitive that the peer MTSE user has refused to accept the multiplex table entry.

If timer T104 expires then the outgoing MTSE user is informed with the REJECT.indication primitive and a MultiplexEntrySendRelease message is sent.

Only MultiplexEntrySendAck and MultiplexEntrySendReject messages which are in response to the most recent MultiplexEntrySend message are accepted. Responses to earlier MultiplexEntrySend messages are ignored.

A new multiplex table entry send request procedure may be initiated with the TRANSFER.request primitive by the user at the outgoing MTSE before a MultiplexEntrySendAck or a MultiplexEntrySendReject message has been received.

C.7.1.2 Protocol overview – Incoming MTSE

When a MultiplexEntrySend message is received at the incoming MTSE, the user is informed of the multiplex table entry send request with the TRANSFER.indication primitive. The incoming MTSE user signals acceptance of the multiplex table entry by issuing the TRANSFER.response primitive,

and a MultiplexEntrySendAck message is sent to the peer outgoing MTSE. The incoming MTSE user signals rejection of the multiplex table entry by issuing the REJECT.request primitive, and a MultiplexEntrySendReject message is sent to the peer outgoing MTSE.

A new MultiplexEntrySend message may be received before the incoming MTSE user has responded to an earlier MultiplexEntrySend message. The incoming MTSE user is informed with the REJECT.indication primitive, followed by the TRANSFER.indication primitive, and the incoming MTSE user responds to the new multiplex table entry.

If a MultiplexEntrySendRelease message is received before the incoming MTSE user has responded to an earlier MultiplexEntrySend message, then the incoming MTSE user is informed with the REJECT.indication, and the earlier multiplex table entry is discarded.

C.7.2 Communication between the MTSE and MTSE user

C.7.2.1 Primitives between MTSE and MTSE user

Communication between the MTSE, and MTSE user, is performed using the primitives shown in Table C.25.

Table C.25/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
TRANSFER	MUX-DESCRIPTOR	MUX-DESCRIPTOR	– (Note 1)	–
REJECT	CAUSE	SOURCE CAUSE	not defined (Note 2)	not defined
NOTE 1 – "–" means no parameters.				
NOTE 2 – "not defined" means that this primitive is not defined.				

C.7.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The TRANSFER primitives are used to transfer multiplex table entries.
- b) The REJECT primitives are used to reject a multiplex table entry, and to terminate a multiplex table entry transfer.

C.7.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.25 is as follows:

- a) The MUX-DESCRIPTOR parameter is a multiplex table entry. This parameter is mapped to the MultiplexEntryDescriptor field of the MultiplexEntrySend message and carried transparently from the MTSE user at the outgoing MTSE to the MTSE user at the incoming MTSE. There may be multiple MUX-DESCRIPTORs associated with the TRANSFER primitive.
- b) The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of "USER" or "PROTOCOL". The latter case may occur as the result of a timer expiry.
- c) The CAUSE parameter indicates the reason for rejection of a multiplex table entry. The CAUSE parameter is not present when the SOURCE parameter indicates "PROTOCOL".

C.7.2.4 MTSE states

The following states are used to specify the allowed sequence of primitives between the MTSE and the MTSE user. The states are specified separately for each of an outgoing MTSE and an incoming MTSE. The states for an outgoing MTSE are:

State 0: IDLE

There is no MTSE transfer in progress. The multiplex table entry may be used by the transmitter.

State 1: AWAITING RESPONSE

The MTSE user has requested the transfer of a multiplex table entry, and a response from the peer MTSE is awaited. The multiplex table entry shall not be used by the transmitter.

The states for an incoming MTSE are:

State 0: IDLE

There is no MTSE transfer in progress. The multiplex table entry may be in use by the transmitter.

State 1: AWAITING RESPONSE

The peer MTSE has transferred a multiplex table entry, and a response from the MTSE user is awaited. The multiplex table entry may not be in use by the transmitter.

C.7.2.5 State transition diagram

The allowed sequence of primitives between the MTSE and the MTSE user is defined here. The allowed sequences are specified separately for each of an outgoing MTSE and an incoming MTSE, as shown in Figure C.25 and Figure C.26 respectively.

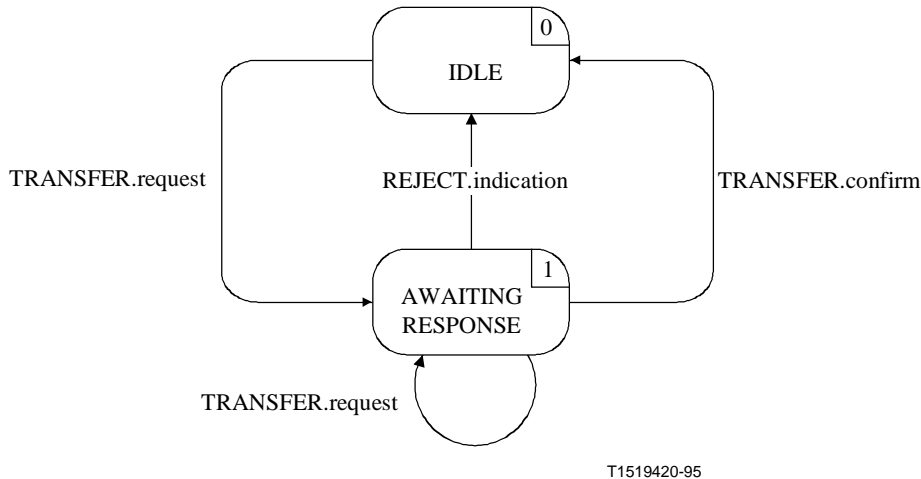


Figure C.25/H.245 – State transition diagram for sequence of primitives at outgoing MTSE

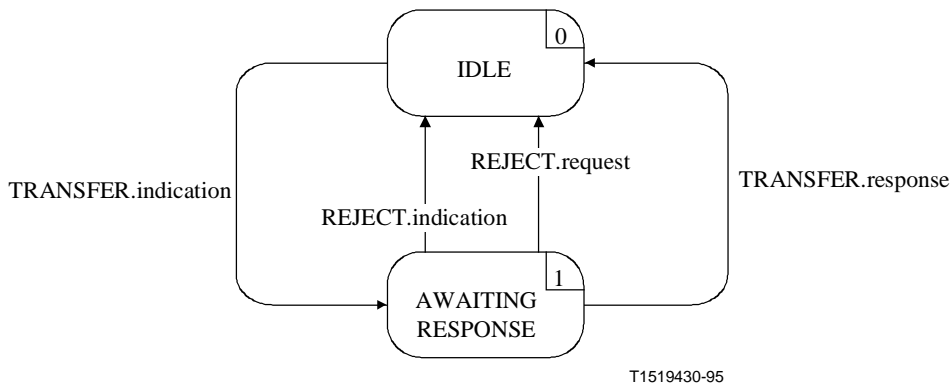


Figure C.26/H.245 – State transition diagram for sequence of primitives at incoming MTSE

C.7.3 Peer-to-peer MTSE communication

C.7.3.1 Messages

Table C.26 shows the MTSE messages and fields, defined in clause 6, which are relevant to the MTSE protocol.

Table C.26/H.245 – MTSE message names and fields

Function	Message	Direction	Field
transfer	MultiplexEntrySend	O → I (Note)	sequenceNumber multiplexEntryDescriptors.multiplexTableEntryNumber multiplexEntryDescriptors.elementList
	MultiplexEntrySendAck	O ← I	sequenceNumber multiplexTableEntryNumber
reject	MultiplexEntrySendReject	O ← I	sequenceNumber multiplexTableEntryNumber rejectionDescriptions.cause
reset	MultiplexEntrySendRelease	O → I	multiplexTableEntryNumber

NOTE – Direction: O – outgoing, I – incoming.

C.7.3.2 MTSE state variables

The following state variables are defined at the outgoing MTSE:

out_ENUM

This state variable distinguishes between outgoing MTSEs. It is initialized at outgoing MTSE initialization. The value of out_ENUM is used to set the multiplexTableEntryNumber field of MTSE messages sent from an outgoing MTSE. For MTSE messages received at an outgoing MTSE, the message multiplexTableEntryNumber field value is identical to the value of out_ENUM.

out_SQ

This state variable is used to indicate the most recently sent MultiplexEntrySend message. It is incremented by one and mapped to the MultiplexEntrySend message sequenceNumber field before transmission of a MultiplexEntrySend message. Arithmetic performed on out_SQ is modulo 256.

The following state variables are defined at the incoming MTSE:

in_ENUM

This state variable distinguishes between incoming MTSEs. It is initialized at incoming MTSE initialization. The value of in_ENUM is used to set the multiplexTableEntryNumber field of MTSE messages sent from an incoming MTSE. For MTSE messages received at an incoming MTSE, the message multiplexTableEntryNumber field value is identical to the value of in_ENUM.

in_SQ

This state variable is used to store the value of the sequenceNumber field of the most recently received MultiplexEntrySend message. The MultiplexEntrySendAck and MultiplexEntrySendReject messages have their sequenceNumber fields set to the value of in_SQ, before being sent to the peer MTSE.

C.7.3.3 MTSE timers

The following timer is specified for the outgoing MTSE:

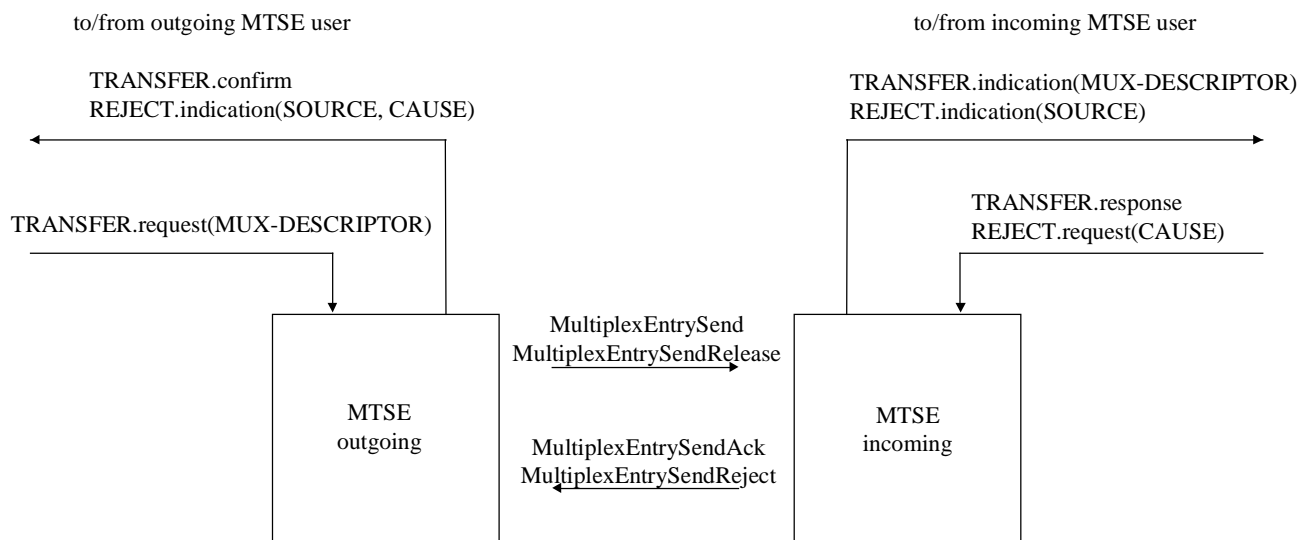
T104

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no MultiplexEntrySendAck or MultiplexEntrySendReject message may be received.

C.7.4 MTSE procedures

C.7.4.1 Introduction

Figure C.27 summarizes the primitives and their parameters, and the messages and relevant fields, for each of the outgoing and incoming MTSE.



T1519440-95

Figure C.27/H.245 – Primitives and messages in the Multiplex Table Signalling Entity

C.7.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.27.

Table C.27/H.245 – Default primitive parameter values

Primitive	Parameter	Default value
TRANSFER.indication	MUX-DESCRIPTOR	MultiplexEntrySend.multiplexEntryDescriptors.elementList
REJECT.indication	SOURCE CAUSE	USER null

C.7.4.3 Message field default values

Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.28.

Table C.28/H.245 – Default message field values

Message	Field	Default value (Note)
MultiplexEntrySend	sequenceNumber	out_SQ
	multiplexEntryDescriptors.multiplexTableEntryNumber	out_ENUM
	multiplexEntryDescriptors.elementList	TRANSFER.request(MUX-DESCRIPTOR)
MultiplexEntrySendAck	sequenceNumber	in_SQ
	multiplexTableEntryNumber	in_ENUM
MultiplexEntrySendReject	sequenceNumber	in_SQ
	rejectionDescriptions.multiplexTableEntryNumber	in_ENUM
	rejectionDescriptions.cause	REJECT.request(CAUSE)
MultiplexEntrySendRelease	multiplexTableEntryNumber	out_ENUM
NOTE – A message field shall not be coded, if the corresponding primitive parameter is null i.e. not present.		

C.7.4.4 SDLs

The outgoing MTSE and the incoming MTSE procedures are expressed in SDL form in Figure C.28 and Figure C.29 respectively.

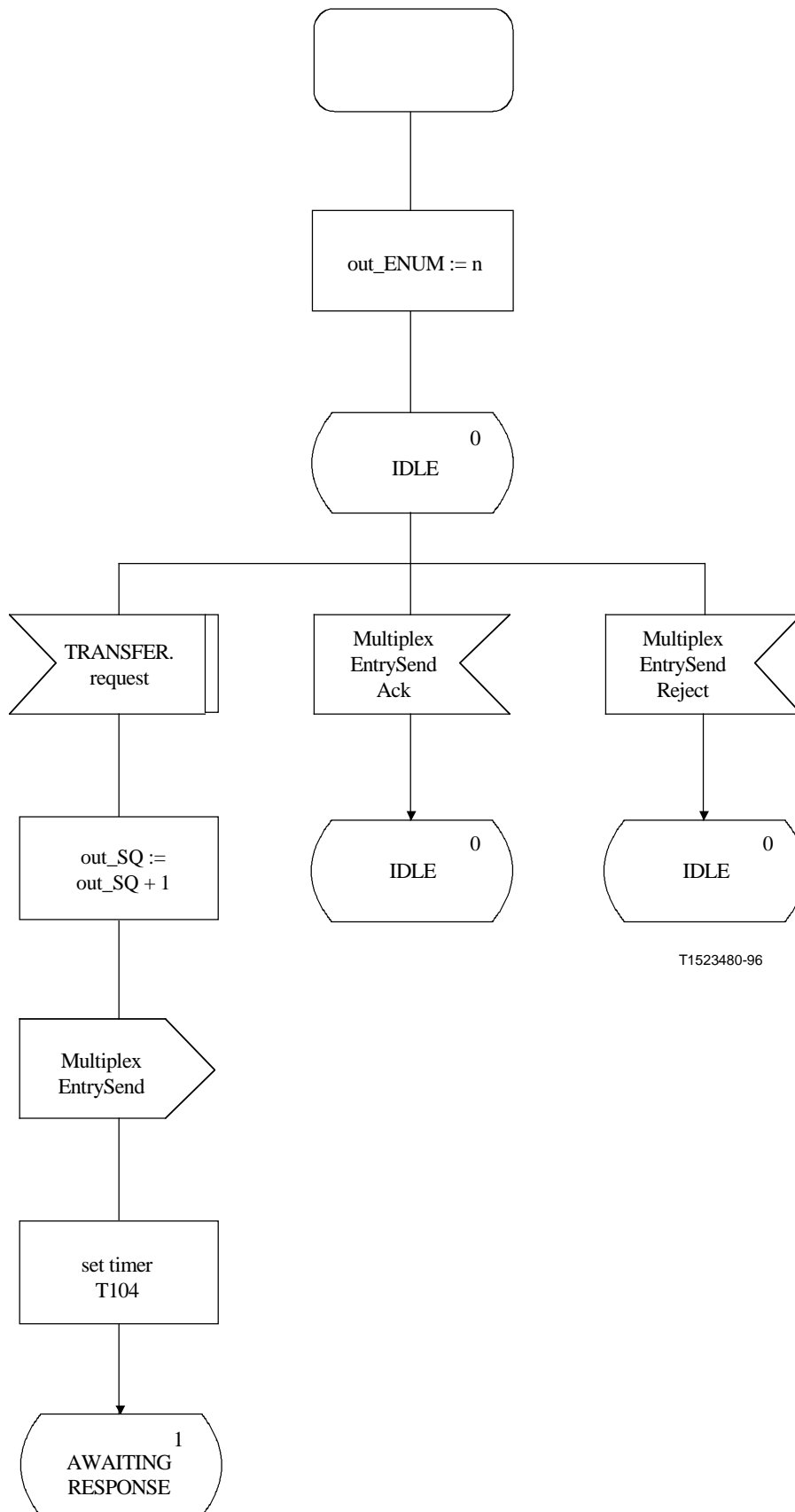


Figure C.28/H.245 – Outgoing MTSE SDL (sheet 1 of 3)

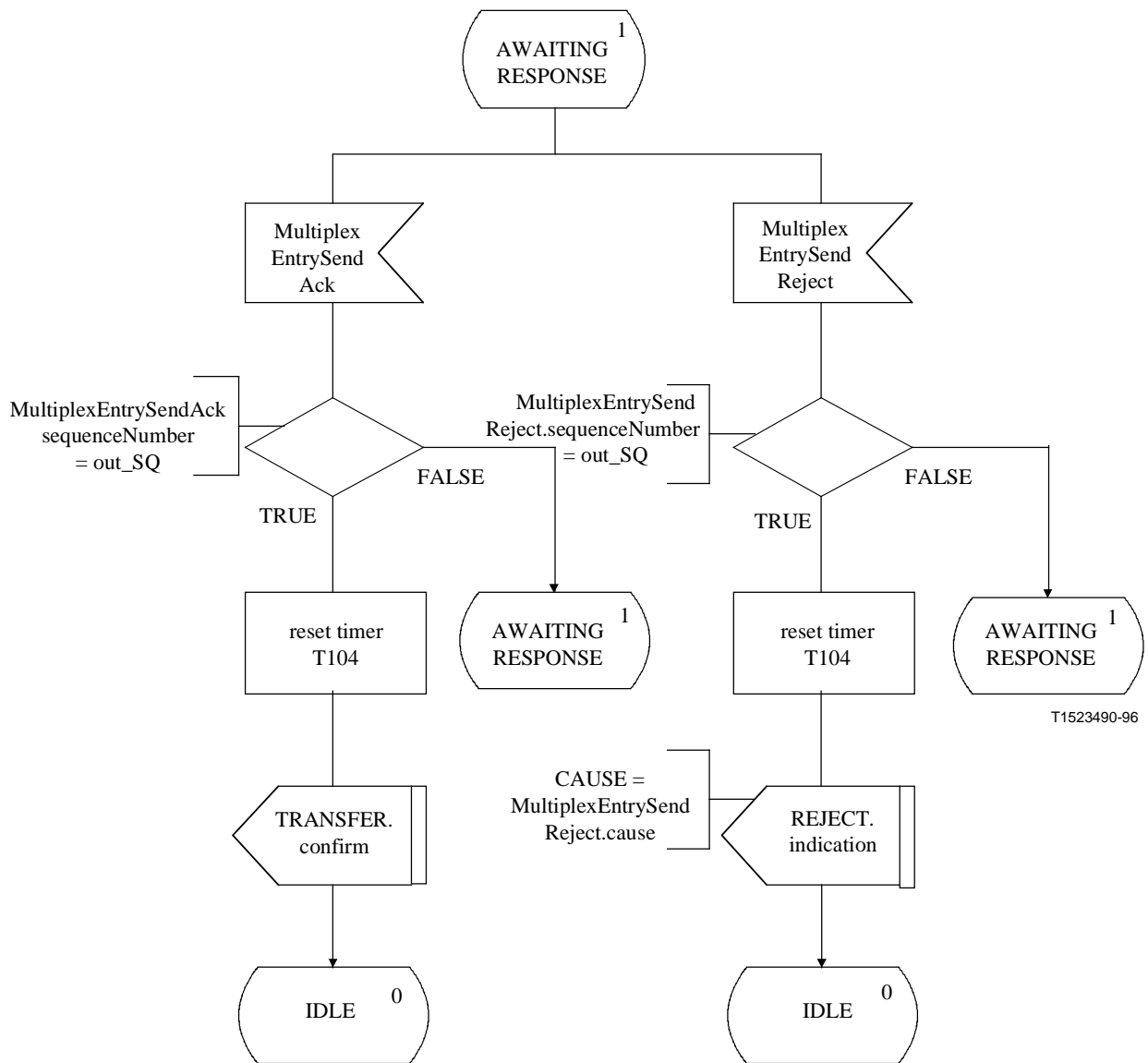


Figure C.28/H.245 – Outgoing MTSE SDL (sheet 2 of 3)

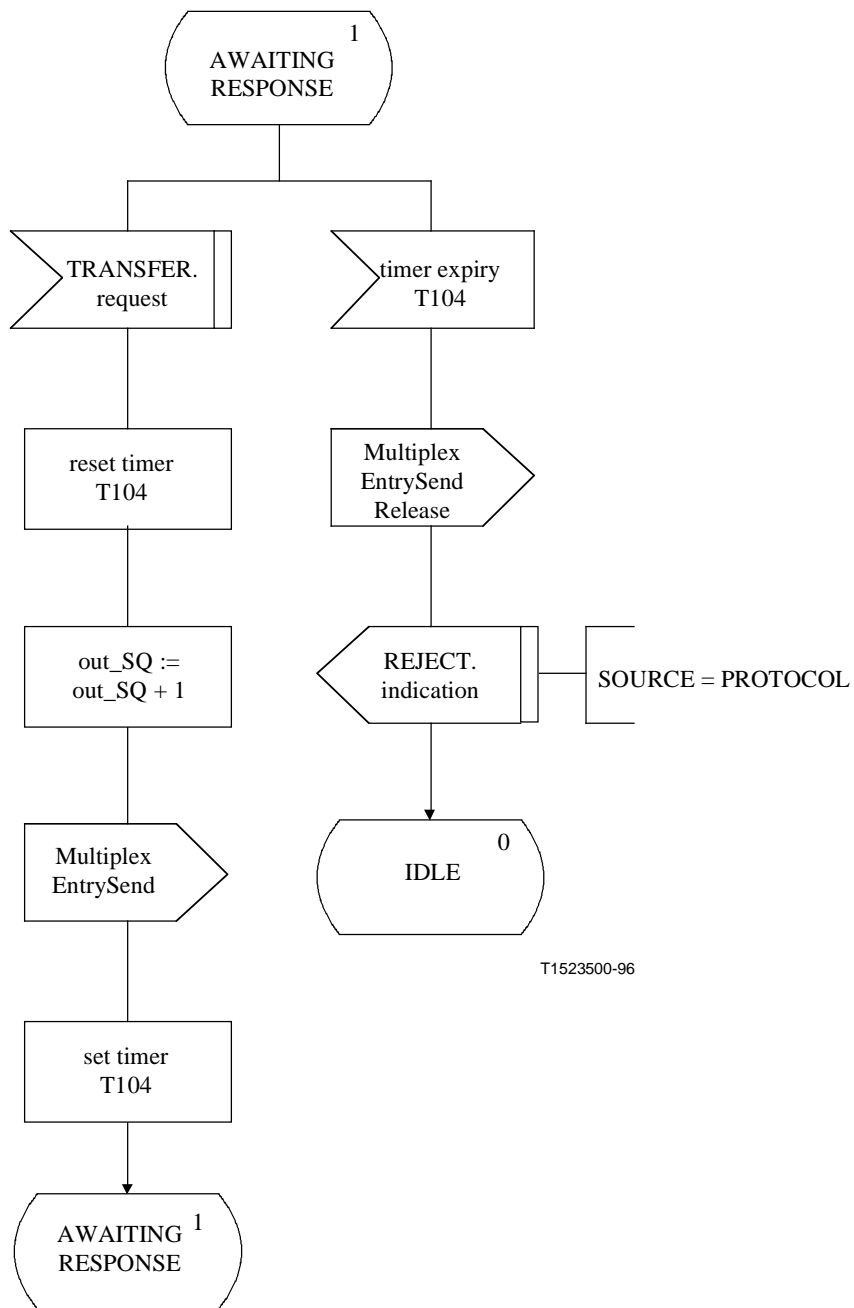


Figure C.28/H.245 – Outgoing MTSE SDL (sheet 3 of 3)

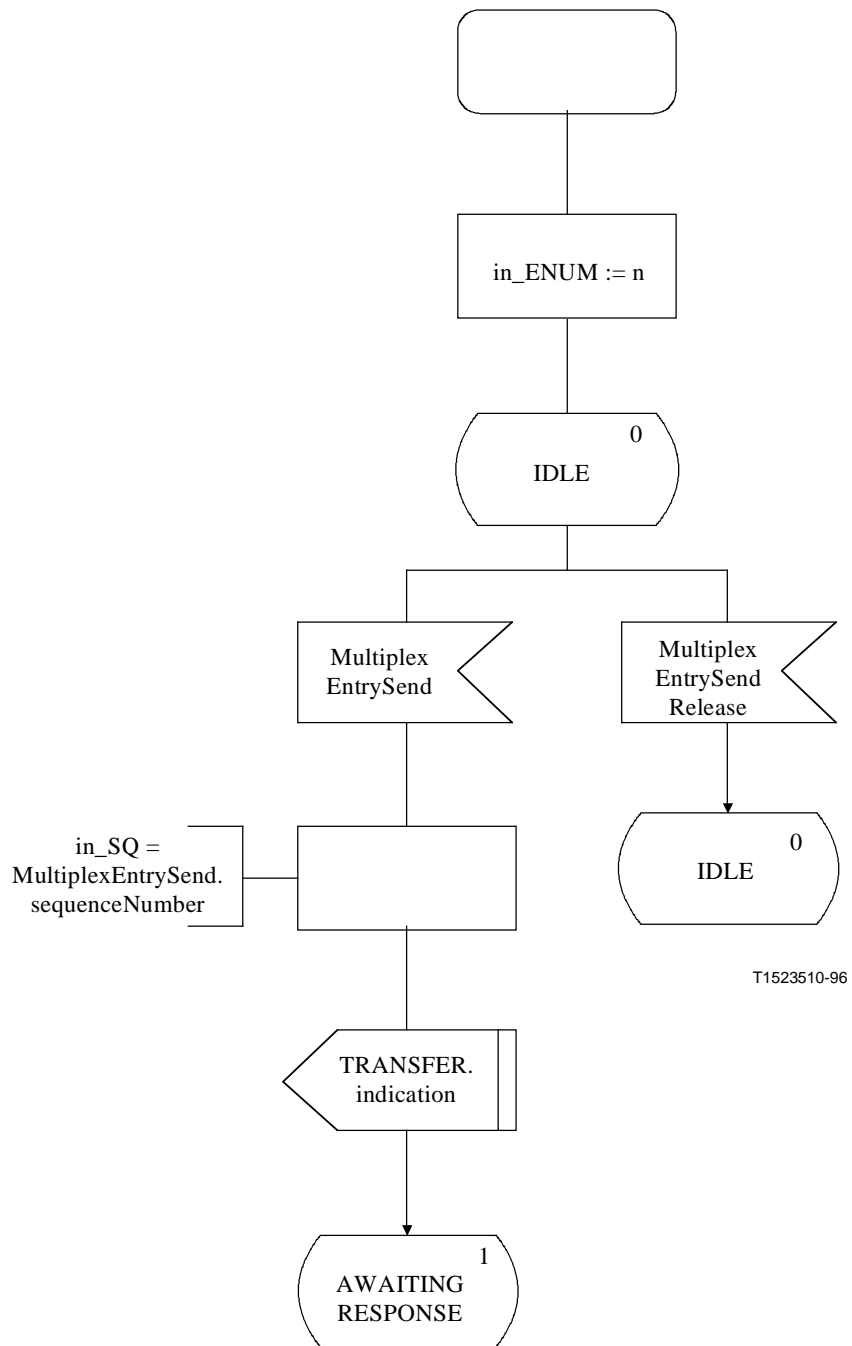


Figure C.29/H.245 – Incoming MTSE SDL (sheet 1 of 2)

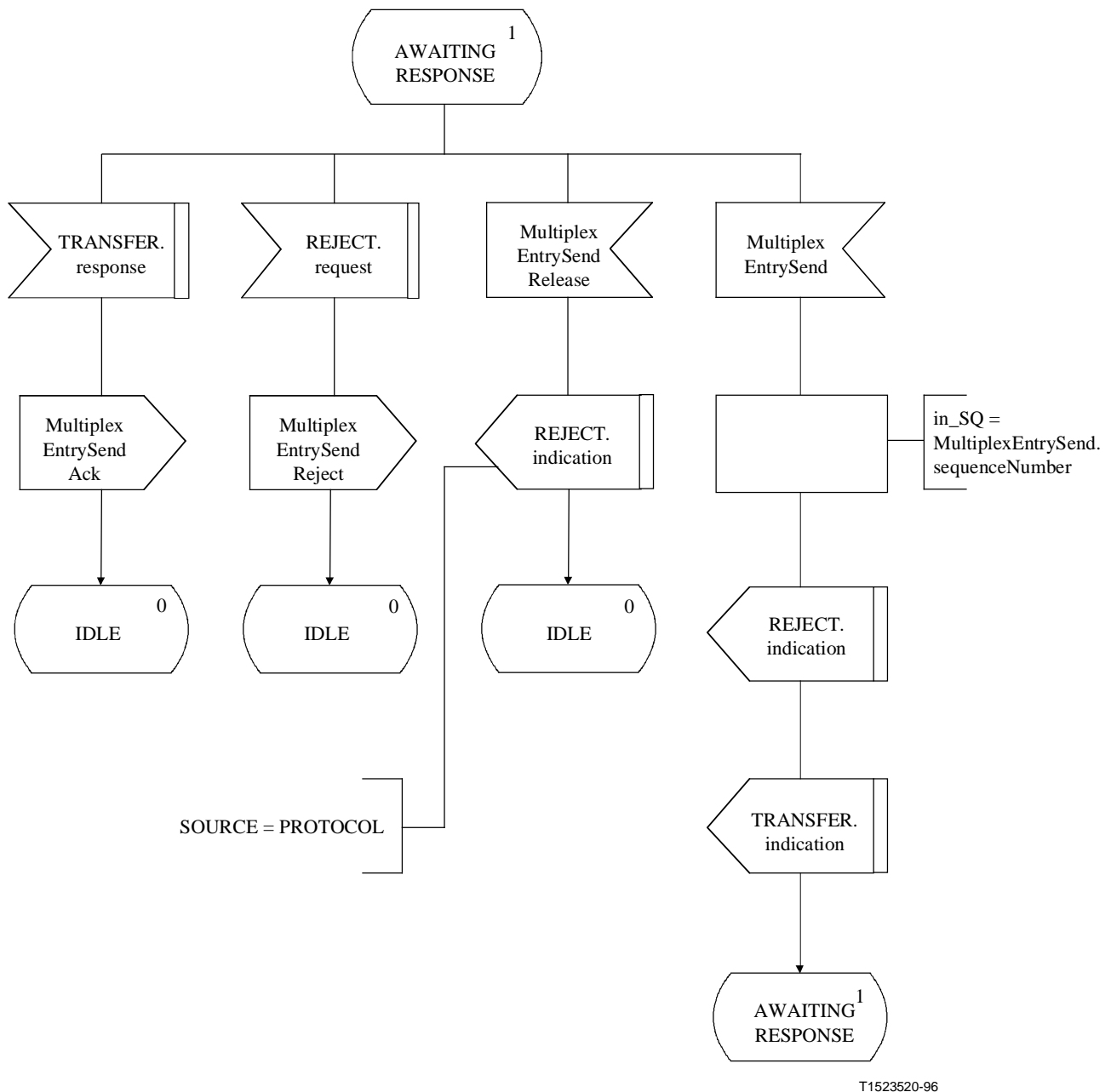


Figure C.29/H.245 – Incoming MTSE SDL (sheet 2 of 2)

C.8 Request Multiplex Entry procedures

C.8.1 Introduction

These procedures are used by a terminal to request the retransmission of one or more MultiplexEntryDescriptors. The procedures are referred to here as the Request Multiplex Entry Signalling Entity (RMESE). Procedures are specified in terms of primitives and states at the interface between the RMESE and the RMESE user. Protocol information is transferred to the peer RMESE via relevant messages defined in clause 6. There is an outgoing RMESE and an incoming RMESE. There is one instance of the RMESE for each multiplex table entry.

A terminal that answers such a response positively, that is, by issuing the SEND.response primitive, shall initiate the Multiplex Table procedures to send the multiplex table entry as soon as possible.

The following text provides an overview of the operation of the protocol. In the case of any discrepancy with the formal specification of the protocol that follows, the formal specification will supersede.

NOTE – This protocol has been defined so that there is an independent RMESE for each multiplex table entry, and the syntax has been defined to allow a single message to carry information relating to one or more multiplex table entries. The way that messages are constructed is an implementation decision: for example, a terminal may respond to a RequestMultiplexEntry message requesting three entries to be sent with one, two or three response messages.

C.8.1.1 Protocol overview – Outgoing RMESE

A request multiplex entry procedure is initiated when the SEND.request primitive is issued by the user at the outgoing RMESE. A RequestMultiplexEntry message is sent to the peer incoming RMESE, and timer T107 is started. If a RequestMultiplexEntryAck message is received in response to the RequestMultiplexEntry message then timer T107 is stopped and the user is informed with the SEND.confirm primitive that the request multiplex entry procedure was successful. If however a RequestMultiplexEntryReject message is received in response to the RequestMultiplexEntry message then timer T107 is stopped and the user is informed with the REJECT.indication primitive that the peer RMESE user has refused to send the multiplex entry.

If timer T107 expires then the outgoing RMESE user is informed with the REJECT.indication primitive and a RequestMultiplexEntryRelease message is sent.

C.8.1.2 Protocol overview – Incoming RMESE

When a RequestMultiplexEntry message is received at the incoming RMESE, the user is informed of the multiplex entry request with the SEND.indication primitive. The incoming RMESE user signals acceptance of the multiplex entry request by issuing the SEND.response primitive, and a RequestMultiplexEntryAck message is sent to the peer outgoing RMESE. The incoming RMESE user signals rejection of the multiplex entry request by issuing the REJECT.request primitive, and a RequestMultiplexEntryReject message is sent to the peer outgoing RMESE.

C.8.2 Communication between RMESE and RMESE user

C.8.2.1 Primitives between RMESE and RMESE user

Communication between the RMESE and RMESE user, is performed using the primitives shown in Table C.29.

Table C.29/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
SEND	– (Note 1)	–	–	–
REJECT	CAUSE	SOURCE CAUSE	not defined (Note 2)	not defined
NOTE 1 – "–" means no parameters.				
NOTE 2 – "not defined" means that this primitive is not defined.				

C.8.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The SEND primitives are used to request the transmission of a multiplex entry.
- b) The REJECT primitives are used to reject the request for transmission of a multiplex entry.

C.8.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.29 are as follows:

- a) The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of "USER" or "PROTOCOL". The latter case may occur as the result of a timer expiry.
- b) The CAUSE parameter indicates the reason for refusal to send a multiplex table entry. The CAUSE parameter is not present when the SOURCE parameter indicates "PROTOCOL".

C.8.2.4 RMESE states

The following states are used to specify the allowed sequence of primitives between the RMESE and the RMESE user.

The states for an outgoing RMESE are:

State 0: IDLE

The RMESE is idle.

State 1: AWAITING RESPONSE

The RMESE is waiting for a response from the remote RMESE.

The states for an incoming RMESE are:

State 0: IDLE

The RMESE is idle.

State 1: AWAITING RESPONSE

The RMESE is waiting for a response from the RMESE user.

C.8.2.5 State transition diagram

The allowed sequence of primitives between the RMESE and the RMESE user is defined here. The allowed sequences are specified separately for each of an outgoing RMESE and an incoming RMESE, as shown in Figure C.30 and Figure C.31 respectively.

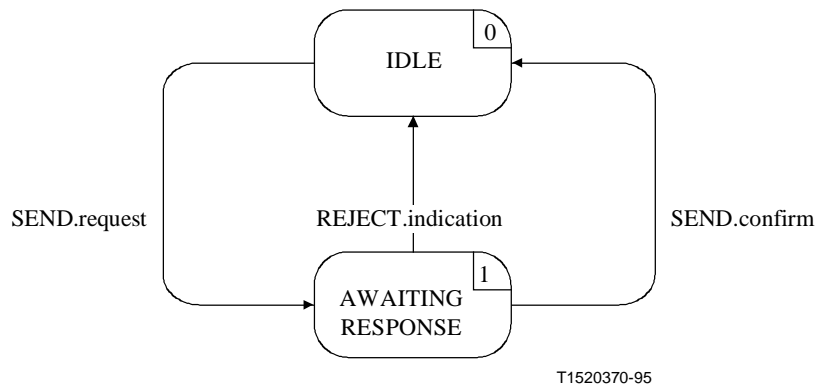


Figure C.30/H.245 – State transition diagram for sequence of primitives at RMESE outgoing

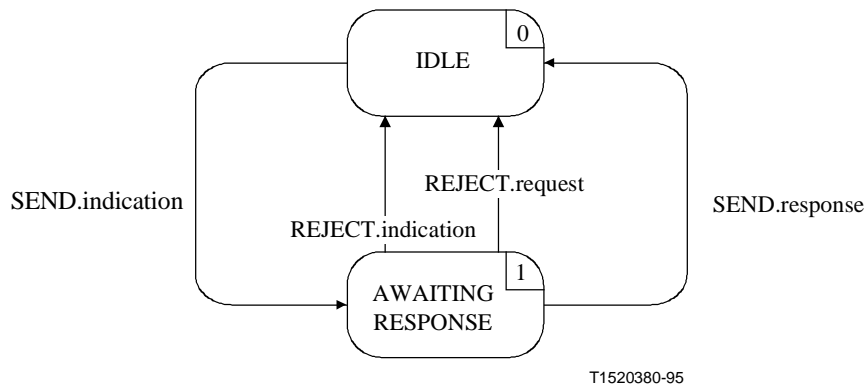


Figure C.31/H.245 – State transition diagram for sequence of primitives at RMESE incoming

C.8.3 Peer-to-peer RMESE communication

C.8.3.1 Messages

Table C.30 shows the RMESE messages and fields, defined in clause 6, which are relevant to the RMESE protocol.

Table C.30/H.245 – RMESE message names and fields

Function	Message	Direction	Field
transfer	RequestMultiplexEntry	O → I (Note)	multiplexTableEntryNumber
	RequestMultiplexEntryAck	O ← I	multiplexTableEntryNumber
	RequestMultiplexEntryReject	O ← I	multiplexTableEntryNumber rejectionDescriptions.cause
reset	RequestMultiplexEntryRelease	O → I	

NOTE – Direction: O – outgoing, I – incoming.

C.8.3.2 RMESE state variables

The following state variable is defined at the outgoing RMESE:

out_ENUM

This state variable distinguishes between outgoing RMESEs. It is initialized at outgoing RMESE initialization. The value of out_ENUM is used to set the multiplexTableEntryNumber field of RMESE messages sent from an outgoing RMESE. For RMESE messages received at an outgoing RMESE, the message multiplexTableEntryNumber field value is identical to the value of out_ENUM.

The following state variable is defined at the incoming RMESE:

in_ENUM

This state variable distinguishes between incoming RMESEs. It is initialized at incoming RMESE initialization. The value of in_ENUM is used to set the multiplexTableEntryNumber field of RMESE messages sent from an incoming RMESE. For RMESE messages received at an incoming RMESE, the message multiplexTableEntryNumber field value is identical to the value of in_ENUM.

C.8.3.3 RMESE timers

The following timer is specified for the outgoing RMESE:

T107

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RequestMultiplexEntryAck or RequestMultiplexEntryReject message may be received.

C.8.4 RMESE procedures

Figure C.32 summarizes the RMESE primitives and their parameters, and messages, for each of the outgoing and incoming RMESE.

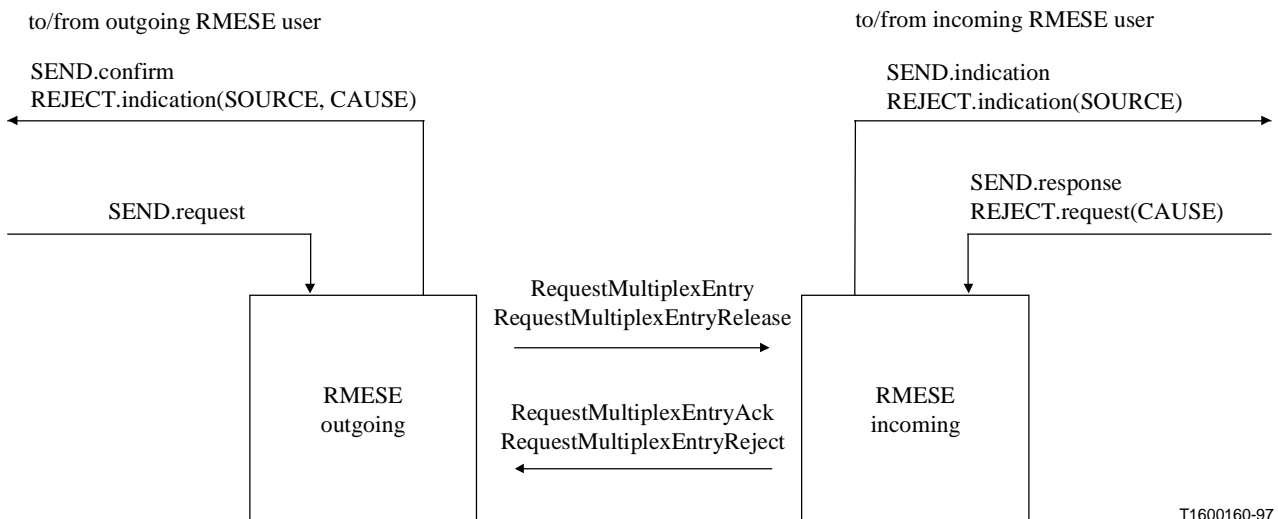


Figure C.32/H.245 – Primitives and messages in the Request Multiplex Entry Signalling Entity

C.8.4.1 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.31.

Table C.31/H.245 – Default primitive parameter values

Primitive	Parameter	Default value
REJECT.indication	SOURCE	USER
	CAUSE	null

C.8.4.2 Message field default values

Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.32.

Table C.32/H.245 – Default message field values

Message	Field	Default value
RequestMultiplexEntry	multiplexTableEntryNumber	out_ENUM
RequestMultiplexEntryAck	multiplexTableEntryNumber	in_ENUM
RequestMultiplexEntryReject	multiplexTableEntryNumber cause	in_ENUM REJECT.request(CAUSE)
RequestMultiplexEntryRelease	multiplexTableEntryNumber	out_ENUM

C.8.4.3 SDLs

The outgoing RMESE and the incoming RMESE procedures are expressed in SDL form in Figure C.33 and Figure C.34 respectively.

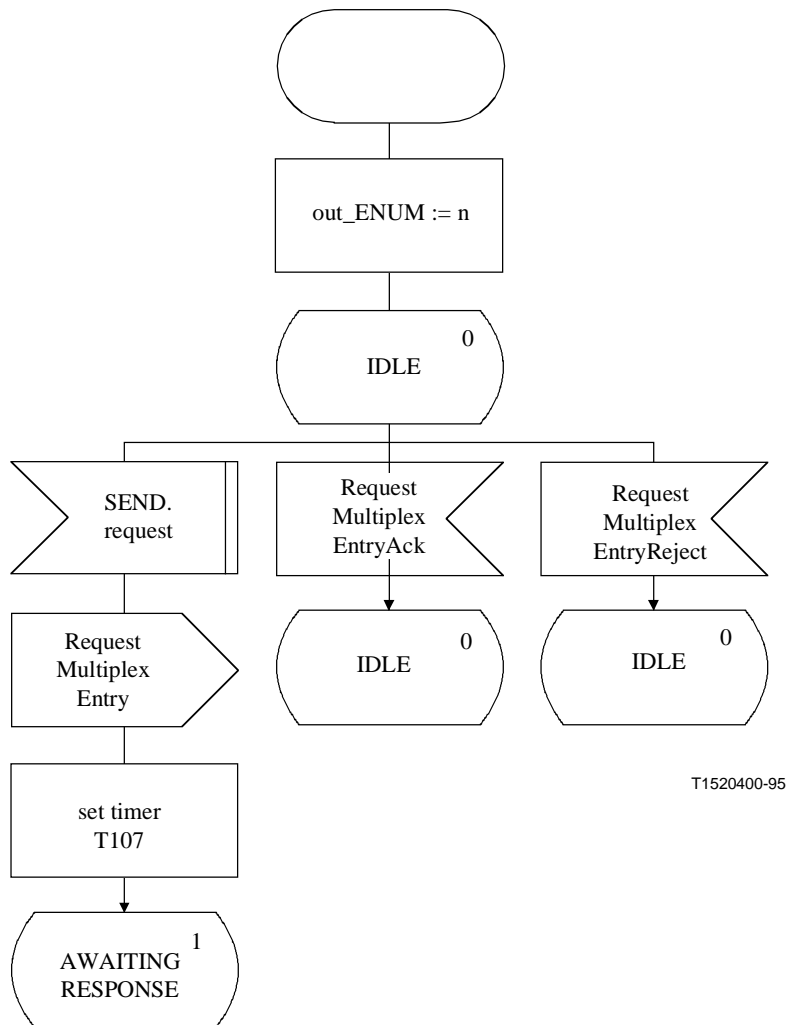


Figure C.33/H.245 – Outgoing RMESE SDL (sheet 1 of 2)

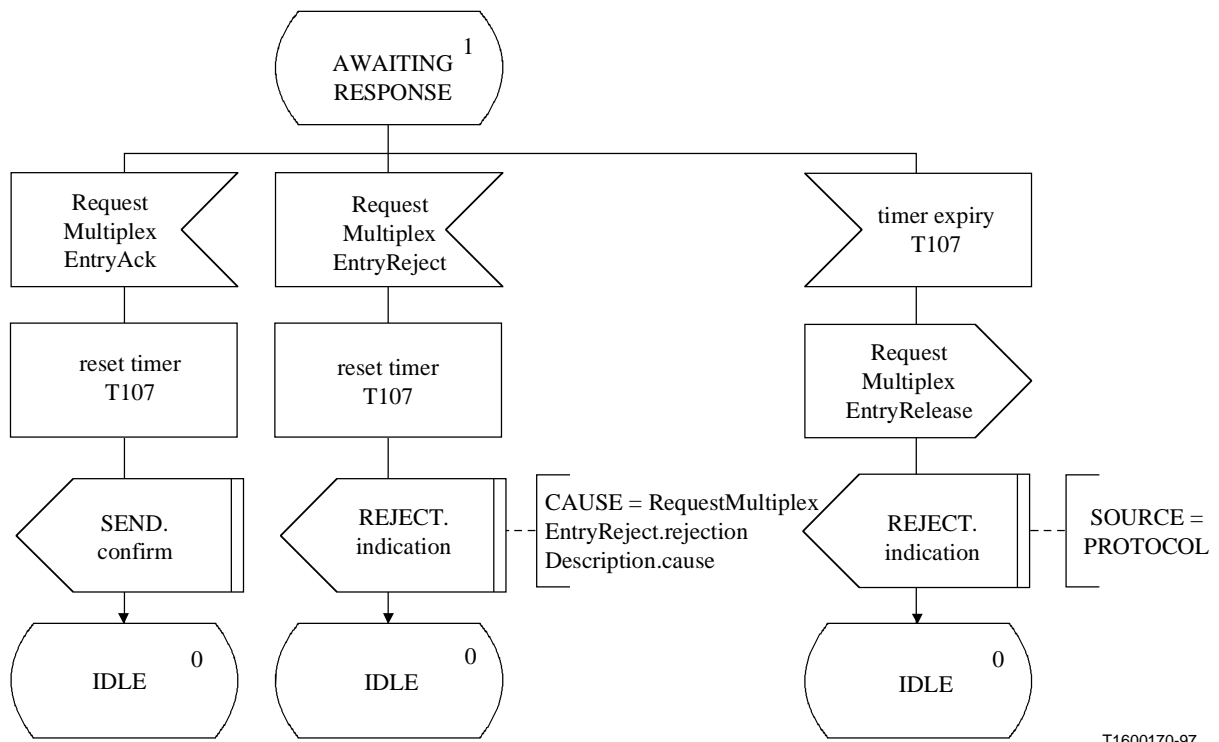


Figure C.33/H.245 – Outgoing RMESE SDL (sheet 2 of 2)

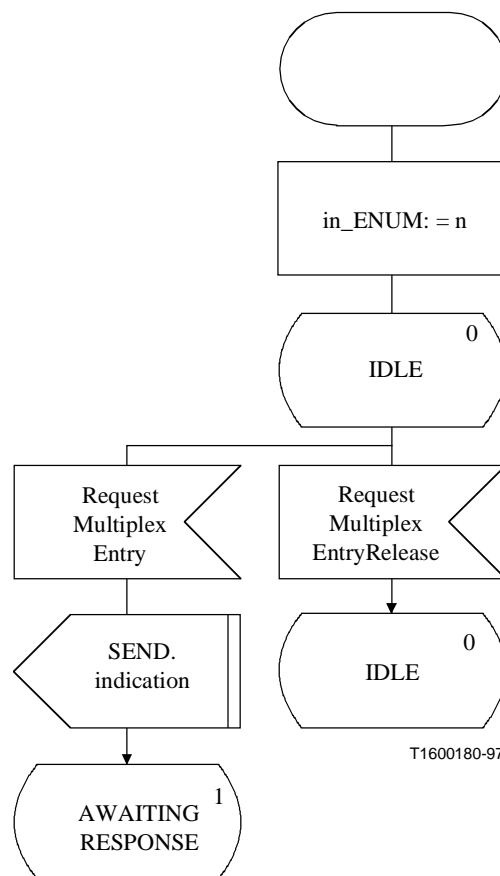


Figure C.34/H.245 – Incoming RMESE SDL (sheet 1 of 2)

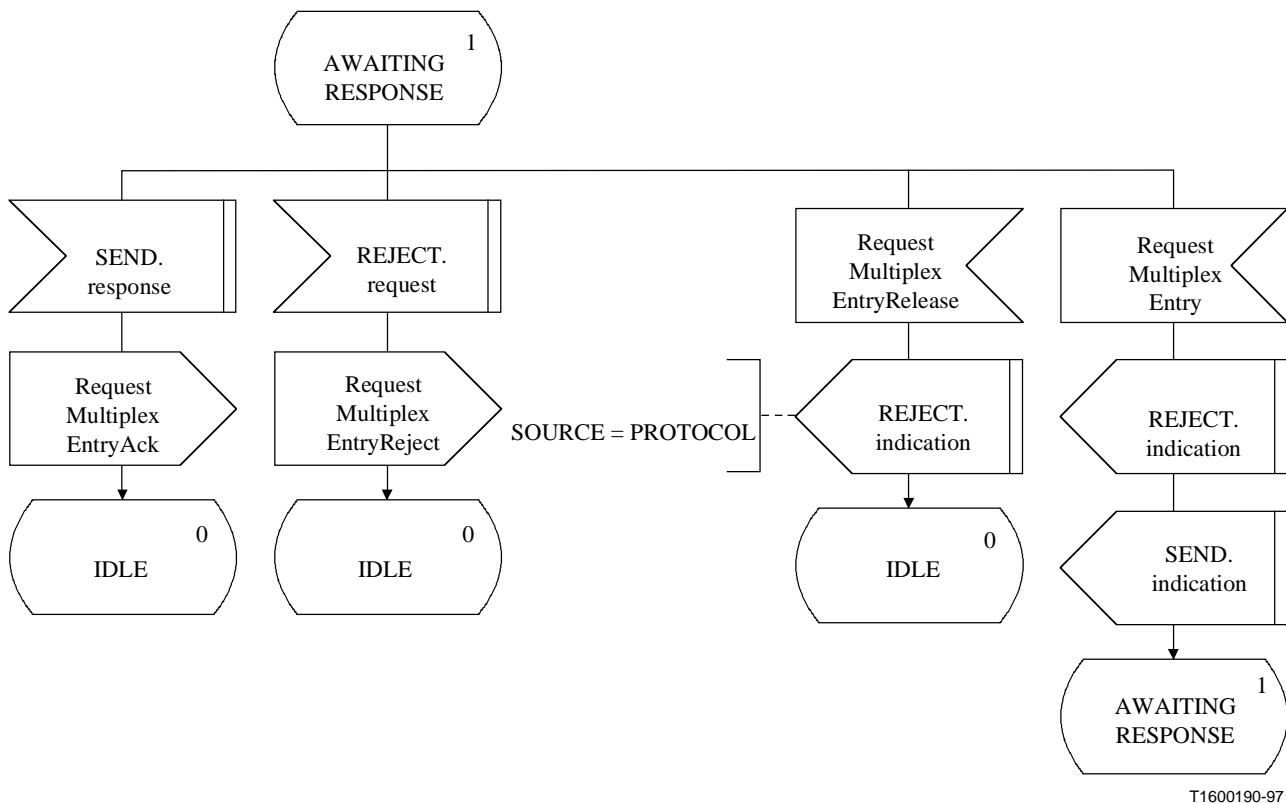


Figure C.34/H.245 – Incoming RMESE SDL (sheet 2 of 2)

C.9 Mode Request procedures

C.9.1 Introduction

The procedures described here allow a terminal to request a remote terminal to use a particular mode of operation in its transmit direction. The procedures are referred to here as the Mode Request Signalling Entity (MRSE). Procedures are specified in terms of primitives and states at the interface between the MRSE and the MRSE user. Protocol information is transferred to the peer MRSE via relevant messages defined in clause 6. There is an outgoing MRSE and an incoming MRSE. At each of the outgoing and incoming ends there is one instance of the MRSE per call.

A terminal that answers such a response positively, that is, by issuing the **TRANSFER.response** primitive, shall initiate the logical channel signalling procedures to establish the appropriate mode of transmission as soon as possible.

If the currently valid capabilities received from the remote terminal contain one or more transmission capabilities, a terminal may select a mode that it prefers to have transmitted to it by performing the Mode Request procedures. A terminal whose currently valid capabilities contain one or more transmission capabilities and which is in receipt of such a request, should comply with the request.

A mode request shall not be sent to a terminal whose currently valid capabilities contain no transmission capabilities, that is, the terminal does not wish to, and shall not, be remotely controlled. If such a terminal does however receive a mode request, it may comply.

A terminal that receives **multipointModeCommand** shall comply with all received mode requests, until the command is cancelled by receipt of **cancelMultipointModeCommand**. A mode request may be sent to a terminal whose currently valid capabilities contain no transmission capabilities when **multipointModeCommand** has previously been sent.

The requested mode may include channels which are already open. For example, if a channel for G.723.1 was currently open and a terminal wished to receive an additional G.728 channel, it would send a mode request containing both the G.723.1 and the G.728 channel. If the G.723.1 channel request were absent, this would indicate that G.723.1 was no longer desired.

NOTE – The request mode description specifies a complete mode. If, for example, video is currently being transmitted and a mode request is received that does not include any specification for video, then this requests video transmission to stop.

Where one source is feeding several receivers it may be unable to respond to any received signals such as requests to transmit in a particular mode.

The following text provides an overview of the operation of the MRSE protocol. In the case of any discrepancy between this and the formal specification, the formal specification will supersede.

C.9.1.1 Protocol overview – Outgoing MRSE

A mode request procedure is initiated when the TRANSFER.request primitive is issued by the user at the outgoing MRSE. A RequestMode message is sent to the peer incoming MRSE, and timer T109 is started. If a RequestModeAck message is received in response to the RequestMode message then timer T109 is stopped and the user is informed with the TRANSFER.confirm primitive that the mode request was successful. If however a RequestModeReject message is received in response to the RequestMode message, then timer T109 is stopped and the user is informed with the REJECT.indication primitive that the peer MRSE user has refused to accept the mode request.

If timer T109 expires then the outgoing MRSE user is informed with the REJECT.indication primitive and a RequestModeRelease message is sent.

Only RequestModeAck and RequestModeReject messages which are in response to the most recent RequestMode message are accepted. Messages in response to earlier RequestMode messages are ignored.

A new mode request procedure may be initiated with the TRANSFER.request primitive by the user at the outgoing MRSE before a RequestModeAck or a RequestModeReject message has been received.

C.9.1.2 Protocol overview – Incoming MRSE

When a RequestMode message is received at the incoming MRSE, the user is informed of the mode request with the TRANSFER.indication primitive. The incoming MRSE user signals acceptance of the mode request by issuing the TRANSFER.response primitive, and a RequestModeAck message is sent to the peer outgoing MRSE. The incoming MRSE user signals rejection of the mode request by issuing the REJECT.request primitive, and a RequestModeReject message is sent to the peer outgoing MRSE.

A new RequestMode message may be received before the incoming MRSE user has responded to an earlier RequestMode message. The incoming MRSE user is informed with the REJECT.indication primitive, followed by the TRANSFER.indication primitive, and the incoming MRSE user responds to the new multiplex table entry.

If a RequestModeRelease message is received before the incoming MRSE user has responded to an earlier RequestMode message, then the incoming MRSE user is informed with the REJECT.indication, and the earlier mode request is discarded.

C.9.2 Communication between MRSE and MRSE user

C.9.2.1 Primitives between MRSE and MRSE user

Communication between the MRSE and MRSE user, is performed using the primitives shown in Table C.33.

Table C.33/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
TRANSFER	MODE-ELEMENT	MODE-ELEMENT	MODE-PREF	MODE-PREF
REJECT	CAUSE	SOURCE CAUSE	not defined (Note)	not defined

NOTE – "not defined" means that this primitive is not defined.

C.9.2.2 Primitive definition

The definition of these primitives is as follows:

- The TRANSFER primitives are used for the transfer of the mode request.
- The REJECT primitives are used to reject a mode request.

C.9.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.33 is as follows:

- The MODE-ELEMENT parameter specifies a mode element. This parameter is mapped to the requestedModes field of the RequestMode message and is carried transparently from the outgoing MRSE user to the incoming MRSE user. This parameter is mandatory. There may be multiple MODE-ELEMENTS associated with the TRANSFER primitives.
- The MODE-PREF parameter informs the user as to whether the most preferred mode requested will be used or not. This parameter is mapped to the response field of the RequestModeAck message and carried transparently from the incoming RMSE user to the outgoing RMSE user. It has two values being "MOST-PREFERRED" and "LESS-PREFERRED".
- The SOURCE parameter indicates the source of the REJECT.indication primitive. The SOURCE parameter has the value of "USER" or "PROTOCOL". The latter case may occur as the result of a timer expiry.
- The CAUSE parameter indicates the reason for refusal to reject a mode request. The CAUSE parameter is not present when the SOURCE parameter indicates "PROTOCOL".

C.9.2.4 MRSE states

The following states are used to specify the allowed sequence of primitives between the MRSE and the MRSE user. The states for an outgoing MRSE are:

State 0: IDLE

The MRSE is idle.

State 1: AWAITING RESPONSE

The MRSE is waiting for a response from the remote MRSE.

The states for an incoming MRSE are:

State 0: IDLE

The MRSE is idle.

State 1: AWAITING RESPONSE

The MRSE is waiting for a response from the MRSE user.

C.9.2.5 State transition diagram

The allowed sequence of primitives between the MRSE and the MRSE user is defined here. The allowed sequences are specified separately for each of an outgoing MRSE and an incoming MRSE, as shown in Figure C.35 and Figure C.36 respectively.

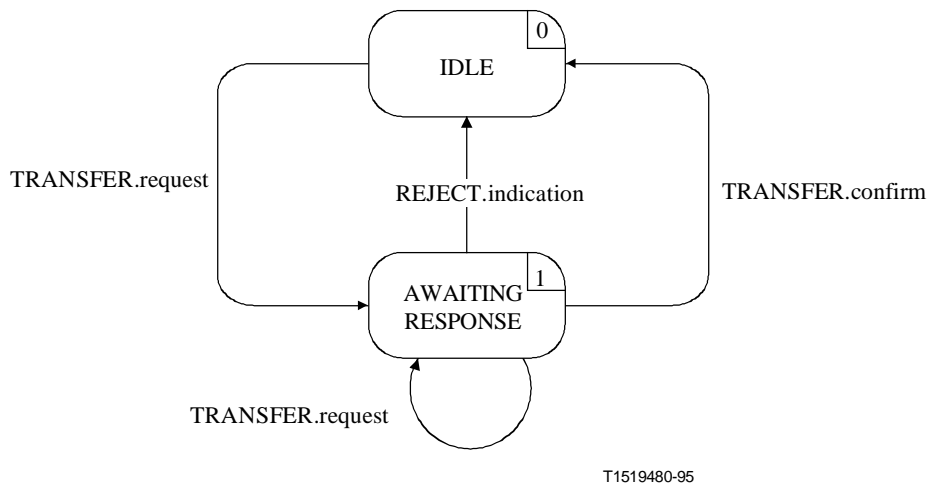


Figure C.35/H.245 – State transition diagram for sequence of primitives at MRSE outgoing

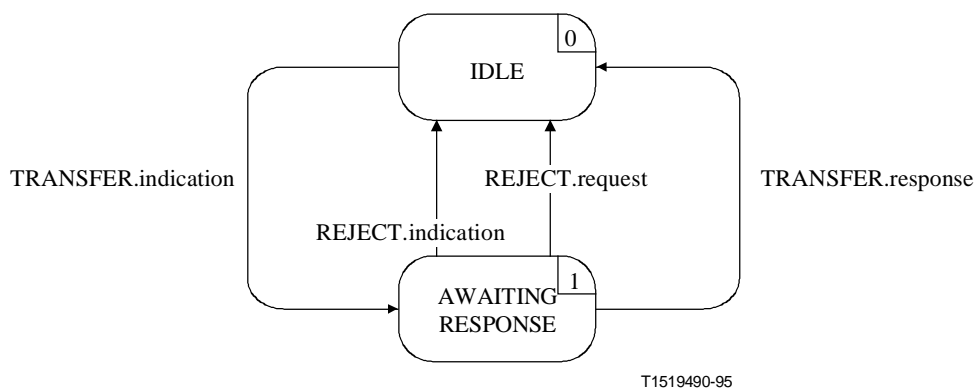


Figure C.36/H.245 – State transition diagram for sequence of primitives at MRSE incoming

C.9.3 Peer-to-peer MRSE communication

C.9.3.1 Messages

Table C.34 shows the MRSE messages and fields, defined in clause 6, which are relevant to the MRSE protocol.

Table C.34/H.245 – MRSE message names and fields

Function	Message	Direction	Field
mode request	RequestMode	O → I (Note)	sequenceNumber requestedModes
	RequestModeAck	O ← I	sequenceNumber response
	RequestModeReject	O ← I	sequenceNumber cause
reset	RequestModeRelease	O → I	–

NOTE – Direction: O – outgoing, I – incoming.

C.9.3.2 MRSE state variables

The following state variables are defined at the outgoing MRSE:

out_SQ

This state variable is used to indicate the most recent RequestMode message. It is incremented by one and mapped to the RequestMode message sequenceNumber field before transmission of the RequestMode message. Arithmetic performed on out_SQ is modulo 256.

The following state variables are defined at the incoming MRSE:

in_SQ

This state variable is used to store the value of the sequenceNumber field of the most recently received RequestMode message. The RequestModeAck and RequestModeReject messages have their sequenceNumber fields set to the value of in_SQ, before being sent to the peer MRSE.

C.9.3.3 MRSE timers

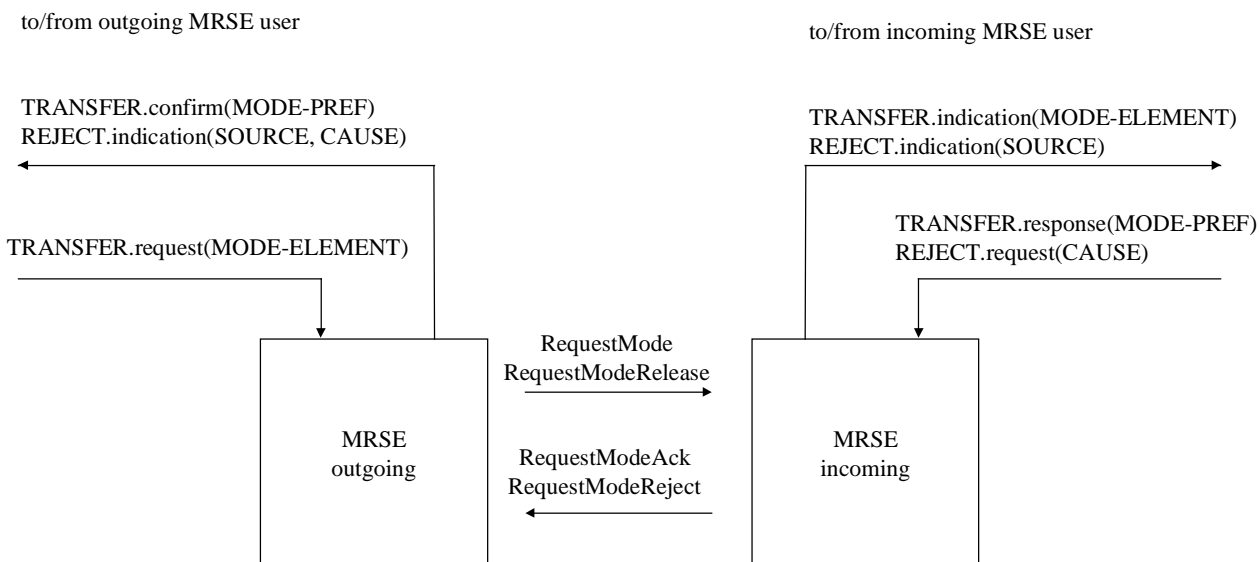
The following timer is specified for the outgoing MRSE:

T109

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RequestModeAck or RequestModeReject message may be received.

C.9.4 MRSE procedures

Figure C.37 summarizes the MRSE primitives and their parameters, and messages, for each of the outgoing and incoming MRSE.



T1519500-95

Figure C.37/H.245 – Primitives and messages in the Mode Request Signalling Entity

C.9.4.1 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.35.

Table C.35/H.245 – Default primitive parameter values

Primitive	Parameter	Default value
TRANSFER.indication	MODE-ELEMENT	RequestMode.requestedModes
TRANSFER.confirm	MODE-PREF	RequestModeAck.response
REJECT.indication	SOURCE	USER
	CAUSE	null

C.9.4.2 Message field default values

Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.36.

Table C.36/H.245 – Default message field values

Message	Field	Default value
RequestMode	sequenceNumber	out_SQ
	requestedModes	TRANSFER.request(MODE-ELEMENT)
RequestModeAck	sequenceNumber	in_SQ
	response	TRANSFER.response(MODE-PREF)
RequestModeReject	sequenceNumber	in_SQ
	cause	REJECT.request(CAUSE)
RequestModeRelease	–	–

C.9.4.3 SDLs

The outgoing MRSE and the incoming MRSE procedures are expressed in SDL form in Figure C.38 and Figure C.39 respectively.

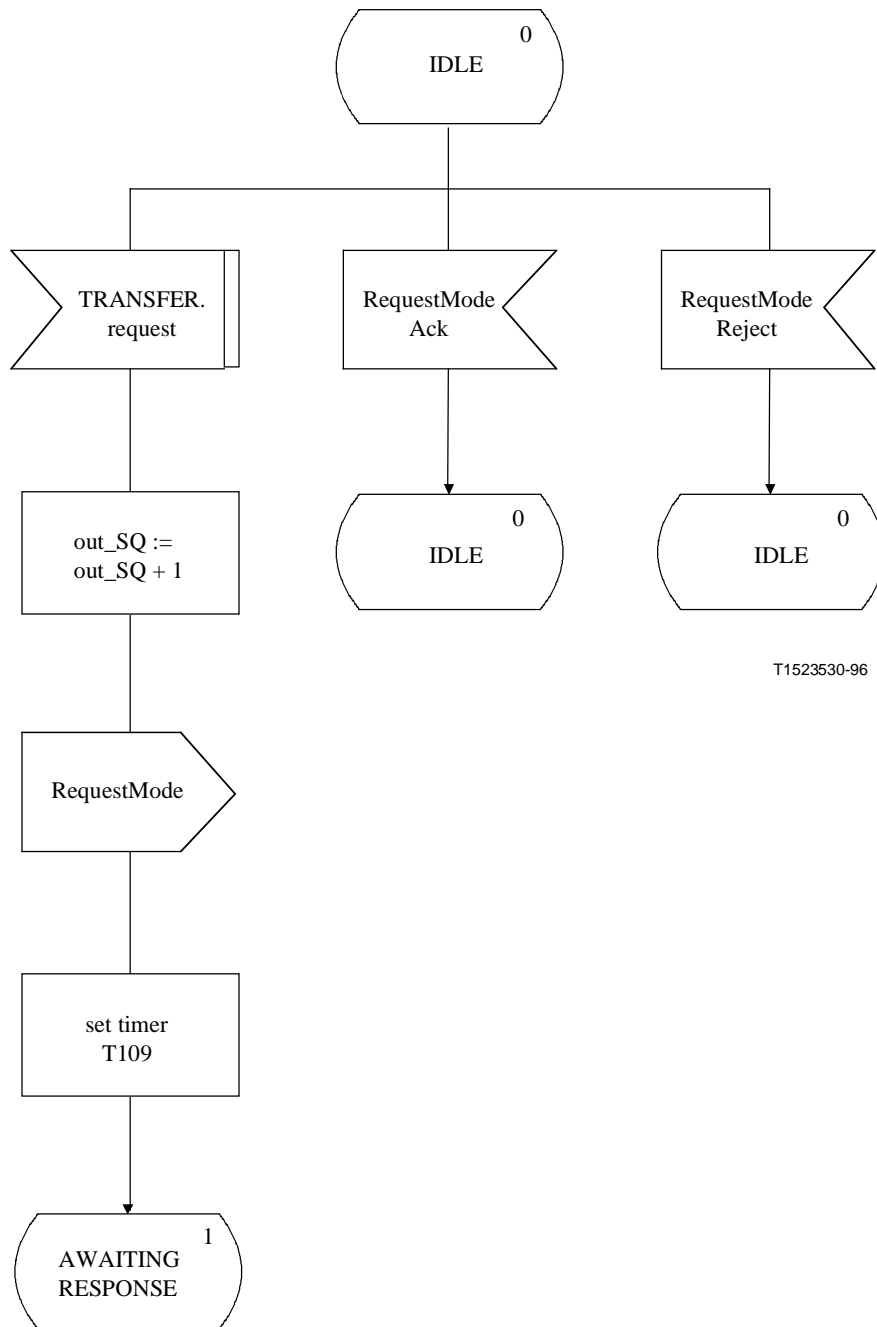


Figure C.38/H.245 – Outgoing MRSE SDL (sheet 1 of 3)

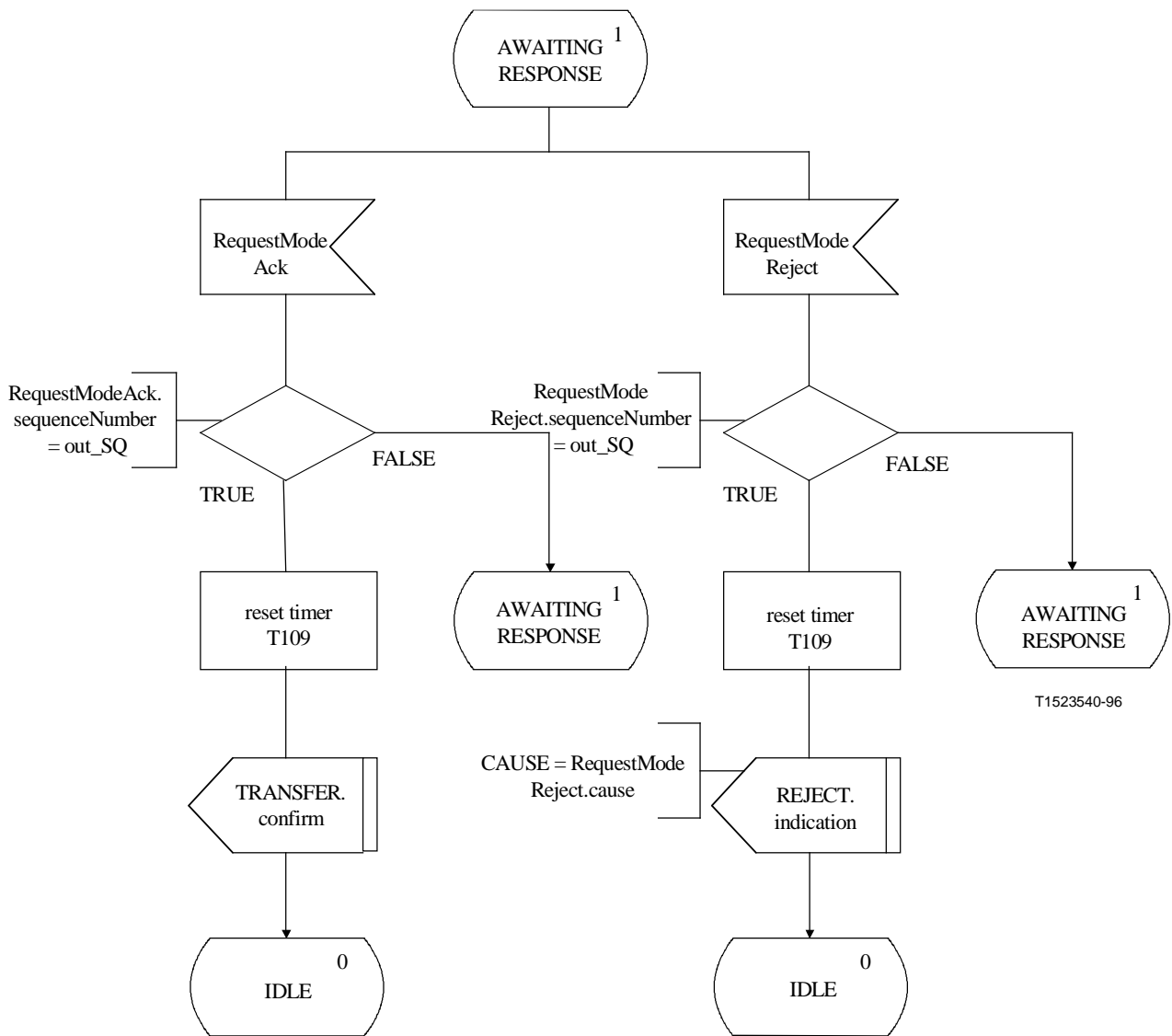


Figure C.38/H.245 – Outgoing MRSE SDL (sheet 2 of 3)

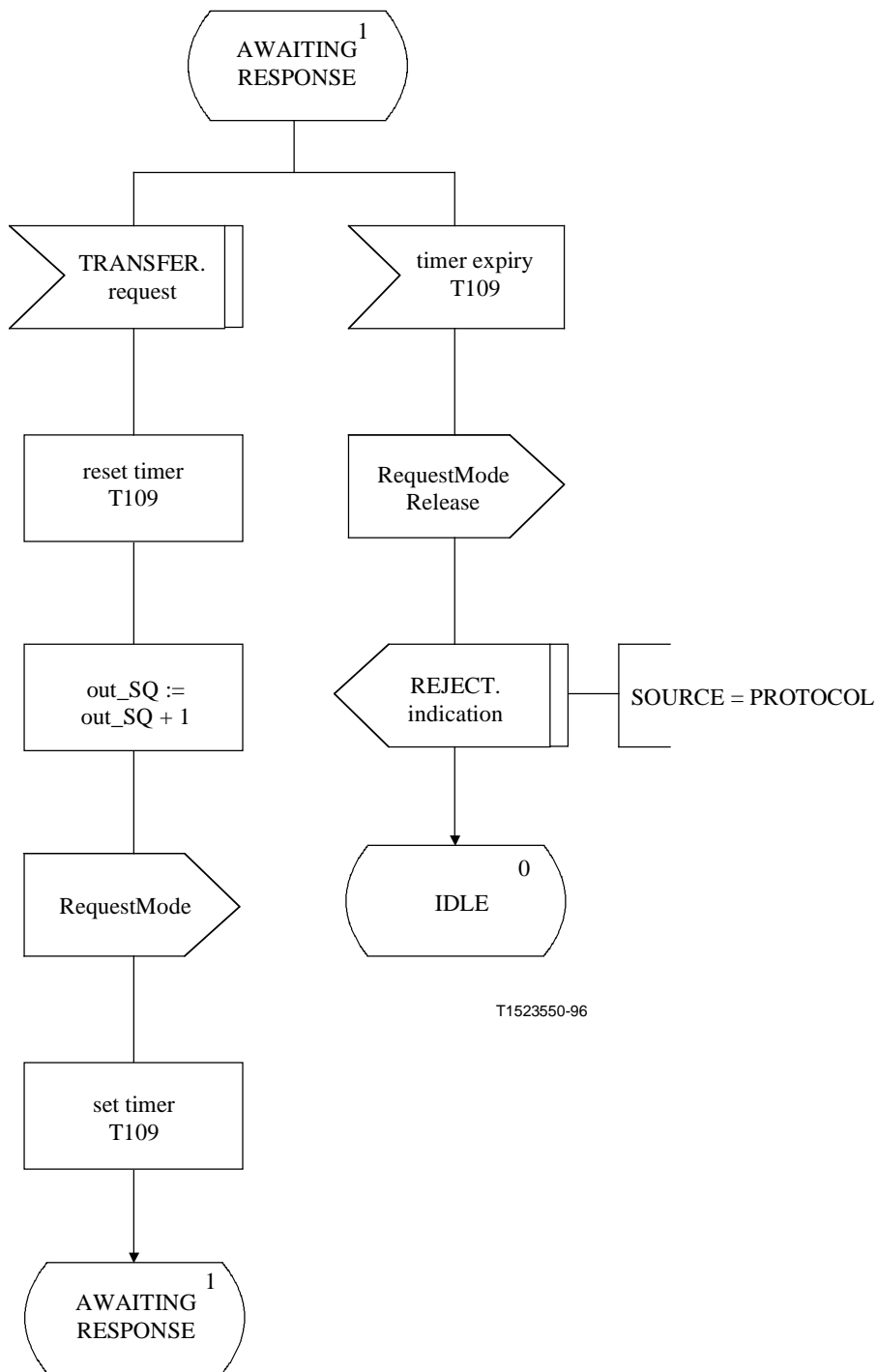


Figure C.38 /H.245 – Outgoing MRSE SDL (sheet 3 of 3)

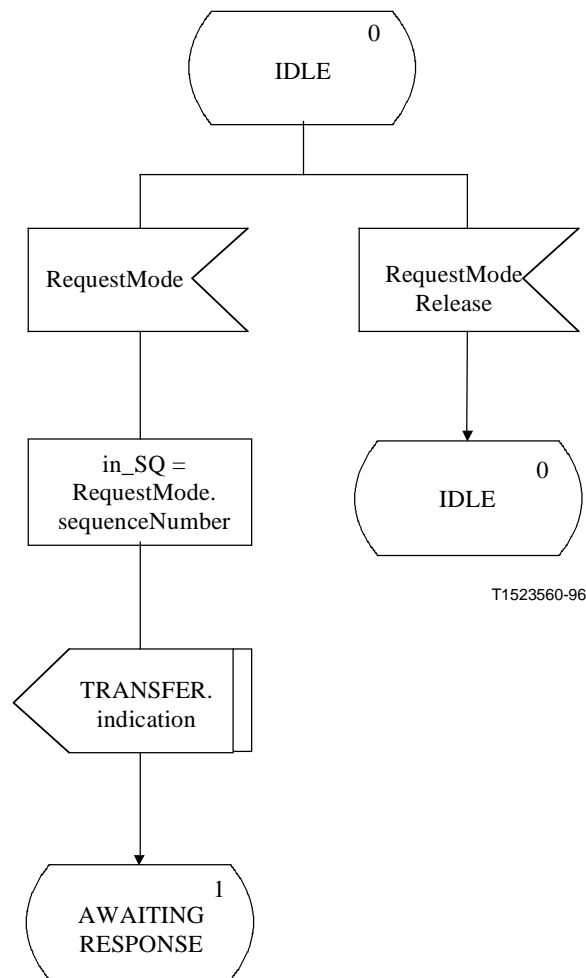
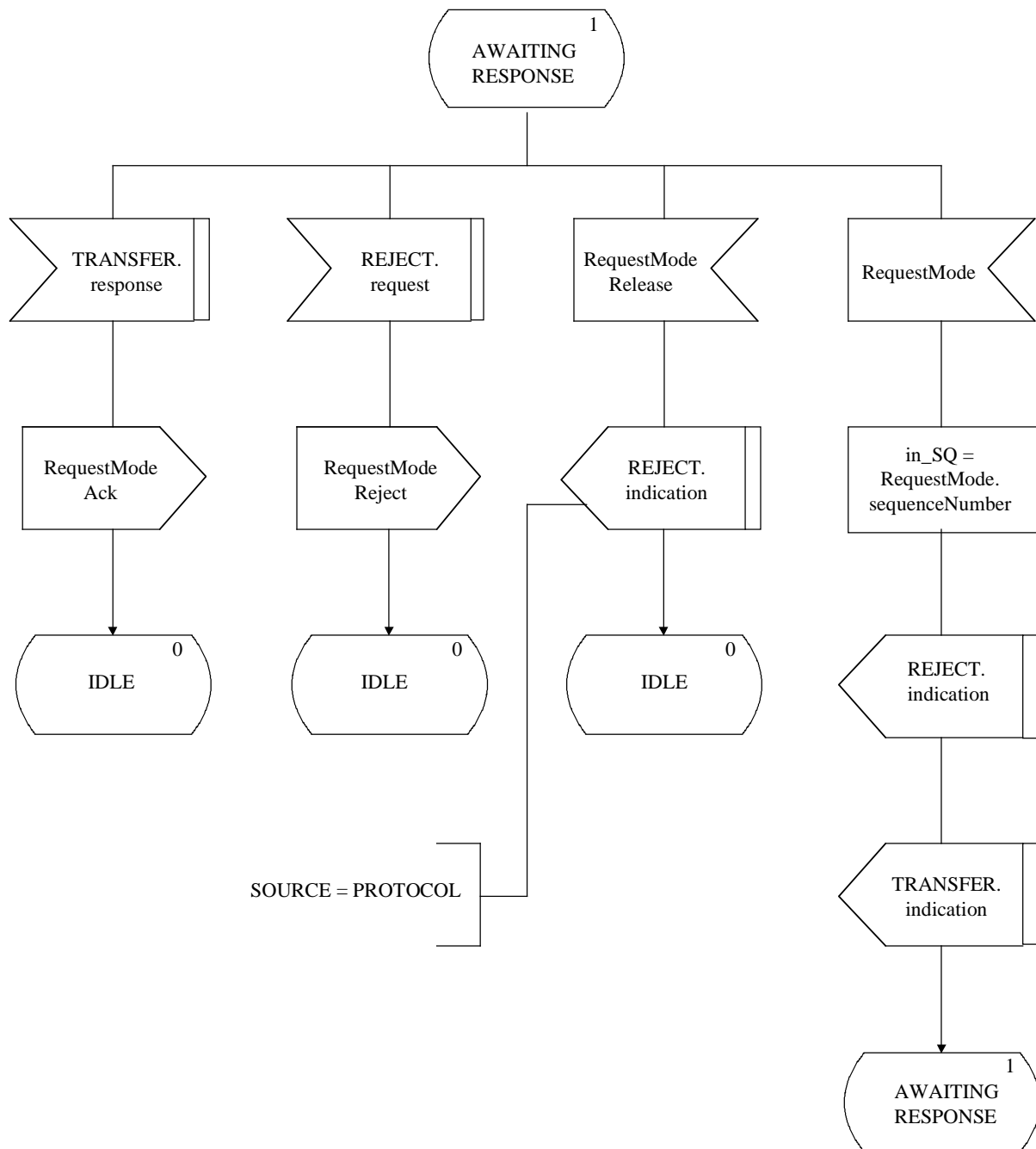


Figure C.39/H.245 – Incoming MRSE SDL (sheet 1 of 2)



T1523570-96

Figure C.39/H.245 – Incoming MRSE SDL (sheet 2 of 2)

C.10 Round trip delay procedures

C.10.1 Introduction

Procedures are described here that allow the determination of the round trip delay between two communicating terminals. This function also enables a H.245 user to determine if the peer H.245 protocol entity is still alive.

The function described here is referred to as the Round Trip Delay Signalling Entity (RTDSE). Procedures are specified in terms of primitives and states at the interface between the RTDSE and the RTDSE user. There is one instance of the RTDSE in each terminal. Any terminal may perform the round trip delay determination.

The following text provides an overview of the operation of the RTDSE protocol. In the case of any discrepancy between this and the formal specification, the formal specification will supersede.

C.10.1.1 Protocol overview – RTDSE

A round trip delay determination procedure is initiated when the TRANSFER.request primitive is issued by the RTDSE user. A RoundTripDelayRequest message is sent to the peer RTDSE, and timer T105 is started. If a RoundTripDelayResponse message is received in response to the RoundTripDelayRequest message then timer T105 is stopped and the user is informed with the TRANSFER.confirm primitive of the round trip delay, which is the value of timer T105.

If a RoundTripDelayRequest message is at any time received from the peer RTDSE, a RoundTripDelayResponse message is immediately sent to the peer RTDSE.

If timer T105 expires then the RTDSE user is informed with the EXPIRY.indication primitive.

Only the RoundTripDelayResponse message which is in response to the most recent RoundTripDelayRequest message is accepted. Messages in response to earlier RoundTripDelayRequest messages are ignored.

A new round trip delay determination procedure may be initiated with the TRANSFER.request primitive by the RTDSE user before a RoundTripDelayResponse message has been received.

C.10.2 Communication between the RTDSE and the RTDSE user

C.10.2.1 Primitives between the RTDSE and the RTDSE user

Communication between the RTDSE and RTDSE user is performed using the primitives shown in Table C.37. These primitives are for the purpose of defining RTDSE procedures and are not meant to specify or constrain implementation.

Table C.37/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
TRANSFER	– (Note 1)	not defined (Note 2)	not defined	DELAY
EXPIRY	not defined	–	not defined	not defined
NOTE 1 – "–" means no parameters.				
NOTE 2 – "not defined" means that this primitive is not defined.				

C.10.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The TRANSFER primitive is used to request, and report upon, the round trip delay determination.
- b) The EXPIRY primitive indicates that no response has been received from the peer terminal.

C.10.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.37 is as follows:

- The DELAY parameter returns the measured round trip delay.

C.10.2.4 RTDSE states

The following states are used to specify the allowed sequence of primitives between the RTDSE and the RTDSE user.

State 0: IDLE:

There is no RTDSE transfer in progress.

State 1: AWAITING RESPONSE:

The RTDSE user has requested the measurement of the round trip delay. A response from the peer RTDSE is awaited.

C.10.2.5 State transition diagram

The allowed sequence of primitives between the RTDSE and the RTDSE user is defined here. The allowed sequences are shown in Figure C.40.

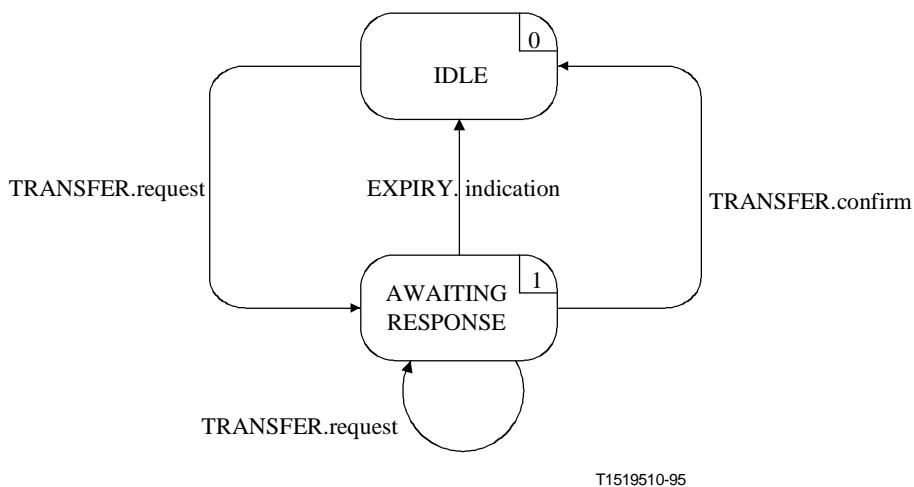


Figure C.40/H.245 – State transition diagram for sequence of primitives at RTDSE

C.10.3 Peer-to-peer RTDSE communication

C.10.3.1 Messages

Table C.38 shows the RTDSE messages and fields, defined in clause 6, which are relevant to the RTDSE protocol.

Table C.38/H.245 – RTDSE message names and fields

Function	Message	Field
transfer	RoundTripDelayRequest	sequenceNumber
	RoundTripDelayResponse	sequenceNumber

C.10.3.2 RTDSE state variables

The following RTDSE state variables are defined:

out_SQ

This state variable is used to indicate the most recent RoundTripDelayRequest message. It is incremented by one and mapped to the RoundTripDelayRequest message sequenceNumber field before transmission of an RoundTripDelayRequest message. Arithmetic performed on out_SQ is modulo 256.

C.10.3.3 RTDSE timers

The following timer is specified for the RTDSE:

T105

This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RoundTripDelayResponse message may be received.

C.10.4 RTDSE procedures

C.10.4.1 Introduction

Figure C.41 summarizes the RTDSE primitives and their parameters, and messages.

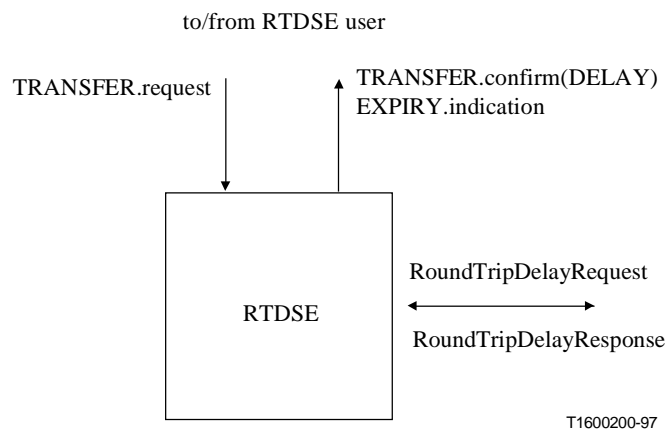


Figure C.41/H.245 – Primitives and messages in the RTDSE

C.10.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.39.

Table C.39/H.245 – Default primitive parameter values

Primitive	Parameter	Default value
TRANSFER.confirm	DELAY	initial value of timer T105 minus value of timer T105
EXPIRY.indication	–	–
NOTE – Timers are defined to count down to zero. The DELAY parameter indicates the time that the timer has been running, and so has the value of the difference between the initial setting and the retained value of the timer.		

C.10.4.3 Message field default values

Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.40.

Table C.40/H.245 – Default message field values

Message	Field	Default value
RoundTripDelayRequest	sequenceNumber	out_SQ
RoundTripDelayResponse	sequenceNumber	RoundTripDelayRequest.sequenceNumber

C.10.4.4 SDLs

The RTDSE procedures are expressed in SDL form in Figure C.42.

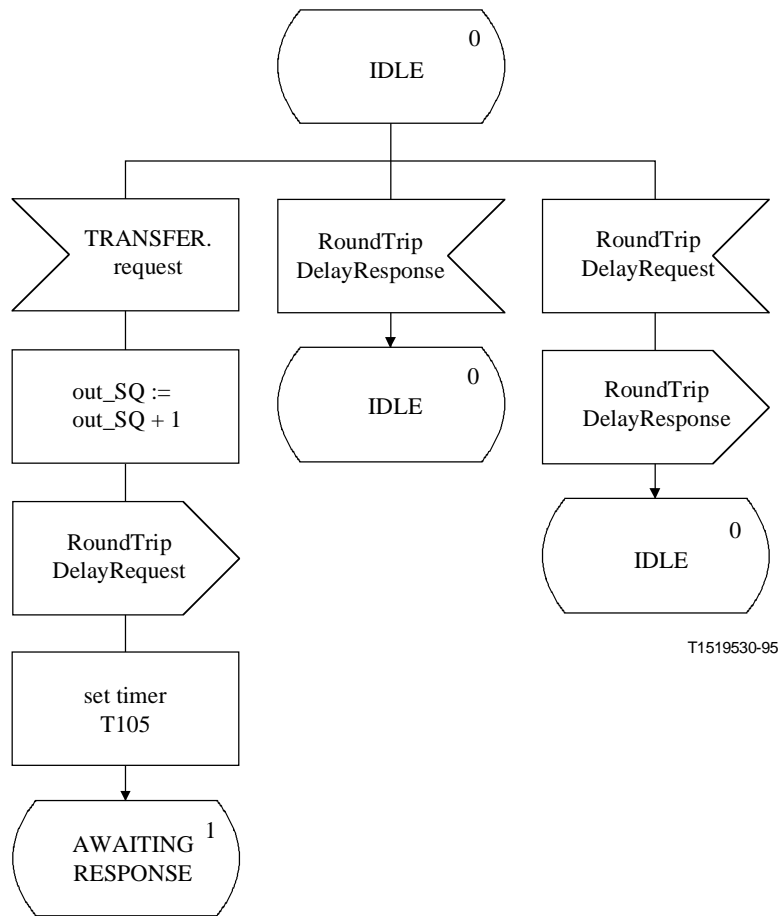


Figure C.42/H.245 – RTDSE SDL (sheet 1 of 2)

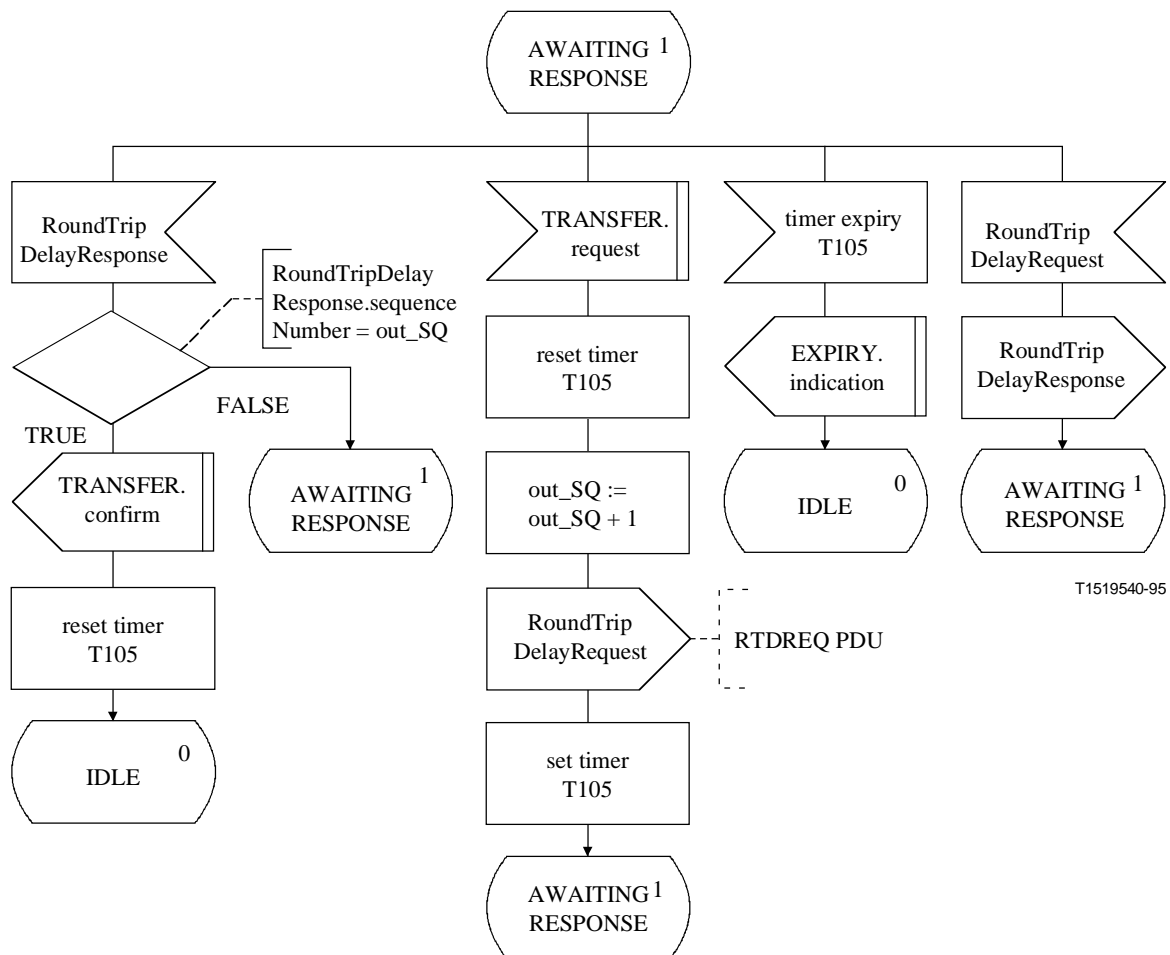


Figure C.42/H.245 – RTDSE SDL (sheet 2 of 2)

C.11 Maintenance Loop procedures

C.11.1 Introduction

The protocol specified here provides reliable operation of maintenance loops using acknowledged procedures.

The protocol specified here is referred to as the Maintenance Loop Signalling Entity (MLSE). Procedures are specified in terms of primitives at the interface between the MLSE and the MLSE user, and MLSE states. Protocol information is transferred to the peer MLSE via relevant messages defined in clause 6.

There is an outgoing MLSE and an incoming MLSE. At each of the outgoing and incoming sides there is one instance of the MLSE for each bidirectional logical channel, and one for the system loop. There is no connection between an incoming MLSE and an outgoing MLSE at one side, other than via primitives to and from the MLSE user. MLSE error conditions are reported.

The terminal that contains the incoming MLSE shall loop the appropriate data while it is in the LOOPED state, and not at any other time. The terminal that contains the outgoing MLSE shall be capable of receiving looped data while in any state, but while in the LOOPED state, should receive looped data only.

NOTE – The MaintenanceLoopOffCommand message applies to all MLSEs. It is always used to stop all maintenance loops.

The following text provides an overview of the operation of the MLSE protocol. In the case of discrepancy between this and the formal specification, the formal specification will supersede.

C.11.1.1 Protocol overview – Outgoing

The establishment of a maintenance loop is initiated when the LOOP.request primitive is issued by the user at the outgoing MLSE. A MaintenanceLoopRequest message is sent to the peer incoming MLSE, and timer T102 is started. If a MaintenanceLoopAck message is received in response to the MaintenanceLoopRequest message, then timer T102 is stopped and the user is informed with the LOOP.confirm primitive that the maintenance loop has been successfully established. If however a MaintenanceLoopReject message is received in response to the MaintenanceLoopRequest message then timer T102 is stopped and the user is informed with the RELEASE.indication primitive that the peer MLSE user has refused establishment of the maintenance loop.

If timer T102 expires in this period then the user is informed with the RELEASE.indication primitive, and a MaintenanceLoopOffCommand message is sent to the peer incoming MLSE. This will cancel all maintenance loops, and not just the one concerned with the particular MLSE.

A maintenance loop that has been successfully established may be cancelled when the RELEASE.request primitive is issued by the user at the outgoing MLSE. A MaintenanceLoopOffCommand message is sent to the peer incoming MLSE.

Before either of the MaintenanceLoopAck or MaintenanceLoopReject messages have been received in response to a previously sent MaintenanceLoopRequest message, the user at the outgoing MLSE may cancel the maintenance loop using the RELEASE.request primitive.

C.11.1.2 Protocol overview – Incoming

When a MaintenanceLoopRequest message is received at the incoming MLSE, the user is informed of the request to establish a maintenance loop with the LOOP.indication primitive. The incoming MLSE user signals acceptance of the request to establish the maintenance loop by issuing the LOOP.response primitive, and a MaintenanceLoopAck message is sent to the peer outgoing MLSE. The maintenance loop shall now be performed. The incoming MLSE user signals rejection of the request to establish the maintenance loop by issuing the RELEASE.request primitive, and a MaintenanceLoopReject message is sent to the peer outgoing MLSE.

A maintenance loop that has been successfully established may be cancelled when the MaintenanceLoopOffCommand message is received at the incoming MLSE. The incoming MLSE user is informed with the RELEASE.indication primitive.

C.11.2 Communication between the MLSE and the MLSE user

C.11.2.1 Primitives between the MLSE and the MLSE user

Communication between the MLSE and the MLSE user is performed using the primitives shown in Table C.41.

Table C.41/H.245 – Primitives and parameters

Generic name	Type			
	request	indication	response	confirm
LOOP	LOOP_TYPE	LOOP_TYPE	– (Note 1)	–
RELEASE	CAUSE	SOURCE CAUSE	not defined (Note 2)	not defined
ERROR	not defined	ERRCODE	not defined	not defined
NOTE 1 – "–" means no parameters.				
NOTE 2 – "not defined" means that this primitive does not exist.				

C.11.2.2 Primitive definition

The definition of these primitives is as follows:

- a) The LOOP primitives are used to establish a maintenance loop.
- b) The RELEASE primitives are used to cancel a maintenance loop.
- c) The ERROR primitive reports MLSE errors to a management entity.

C.11.2.3 Parameter definition

The definition of the primitive parameters shown in Table C.41 is as follows:

- a) The LOOP_TYPE parameter specifies the parameters associated with the maintenance loop. It has values of "SYSTEM", "MEDIA", and "LOGICAL_CHANNEL". This parameter, and the logical channel number, determine the value of the type field of the MaintenanceLoopRequest message which is then carried transparently to the peer MLSE user.
- b) The SOURCE parameter indicates to the MLSE user the source of the maintenance loop release. The SOURCE parameter has the value of "USER" or "MLSE", indicating either the MLSE user, or the MLSE. The latter may occur as the result of a protocol error.
- c) The CAUSE parameter indicates the reason as to why the peer MLSE user rejected a request to establish a maintenance loop. The CAUSE parameter is not present when the SOURCE parameter indicates "MLSE".
- d) The ERRCODE parameter indicates the type of MLSE error. Table C.45 shows the allowed values of the ERRCODE parameter.

C.11.2.4 MLSE states

The following states are used to specify the allowed sequence of primitives between the MLSE and the MLSE user, and the exchange of messages between peer MLSEs. The states are specified separately for each of an outgoing MLSE and an incoming MLSE. The states for an outgoing MLSE are:

State 0: NOT LOOPED

There is no maintenance loop.

State 1: AWAITING RESPONSE

The outgoing MLSE is waiting to establish a maintenance loop with a peer incoming MLSE.

State 2: LOOPED

The MLSE peer-to-peer maintenance loop has been established. All data received on the appropriate channel should be looped data.

The states for an incoming MLSE are:

State 0: NOT LOOPED

There is no maintenance loop.

State 1: AWAITING RESPONSE

The incoming MLSE is waiting to establish a maintenance loop with a peer outgoing MLSE. The appropriate data shall not be looped.

State 2: LOOPED

An MLSE peer-to-peer maintenance loop has been established. All data received on the appropriate channel shall be looped.

C.11.2.5 State transition diagram

The allowed sequence of primitives between the MLSE and the MLSE user is defined here. The allowed sequence of primitives relates to states of the MLSE as viewed from the MLSE user. The allowed sequences are specified separately for each of an outgoing MLSE and an incoming MLSE, as shown in Figure C.43 and Figure C.44 respectively.

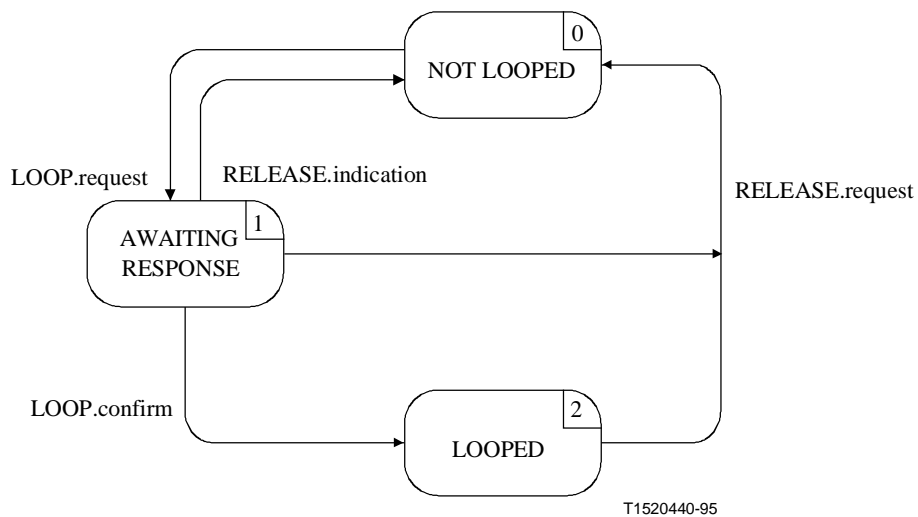


Figure C.43/H.245 – State transition diagram for sequence of primitives at outgoing MLSE

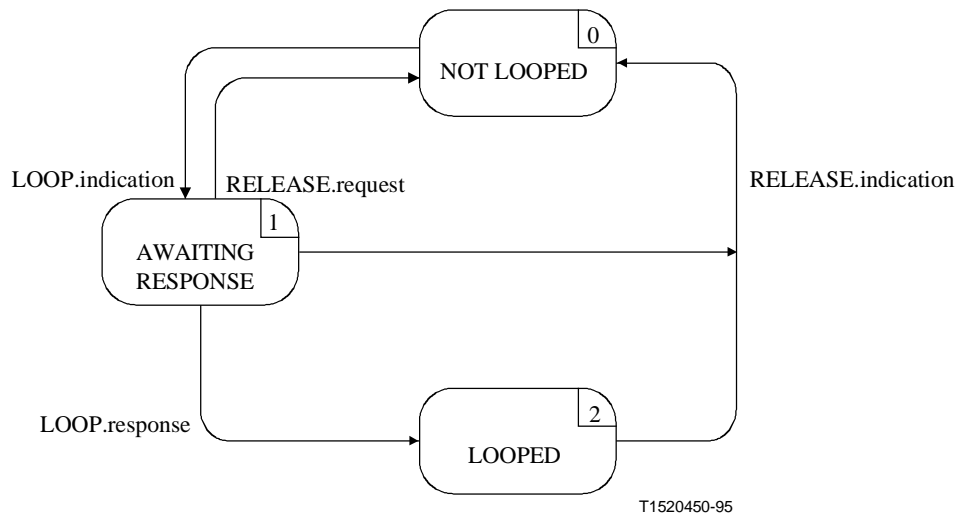


Figure C.44/H.245 – State transition diagram for sequence of primitives at incoming MLSE

C.11.3 Peer-to-peer MLSE communication

C.11.3.1 MLSE messages

Table C.42 shows the MLSE messages and fields, defined in clause 6, which are relevant to the MLSE protocol.

Table C.42/H.245 – MLSE message names and fields

Function	Message	Direction	Field
establish	MaintenanceLoopRequest	O → I (Note)	type
	MaintenanceLoopAck	O ← I	type
	MaintenanceLoopReject		O ← I
			cause
release	MaintenanceLoopOffCommand	O → I	–
NOTE – Direction: O – outgoing, I – incoming.			

C.11.3.2 MLSE state variables

The following state variable is defined at the outgoing MLSE:

out_MLN

This state variable distinguishes between outgoing MLSEs. It is initialized at outgoing MLSE initialization. The value of out_MLN is used to set the type field of MaintenanceLoopRequest messages sent from an outgoing MLSE.

The following state variable is defined at the incoming MLSE:

in_MLN

This state variable distinguishes between incoming MLSEs. It is initialized at incoming MLSE initialization. For MaintenanceLoopRequest messages received at an incoming MLSE, the message type field value is consistent with the value of in_MLN.

in_TYPE

This state variable stores the value of LOOP_TYPE when the MaintenanceLoopRequest is received. This state variable assists in setting the value of the type field in the MaintenanceLoopAck message.

C.11.3.3 MLSE timers

The following timer is specified for the outgoing MLSE:

T102

This timer is used during the AWAITING RESPONSE state. It specifies the maximum allowed time during which no MaintenanceLoopAck or MaintenanceLoopReject message may be received.

C.11.4 MLSE procedures

C.11.4.1 Introduction

Figure C.45 summarizes the primitives and their parameters, and the messages, for each of the outgoing and incoming MLSE.

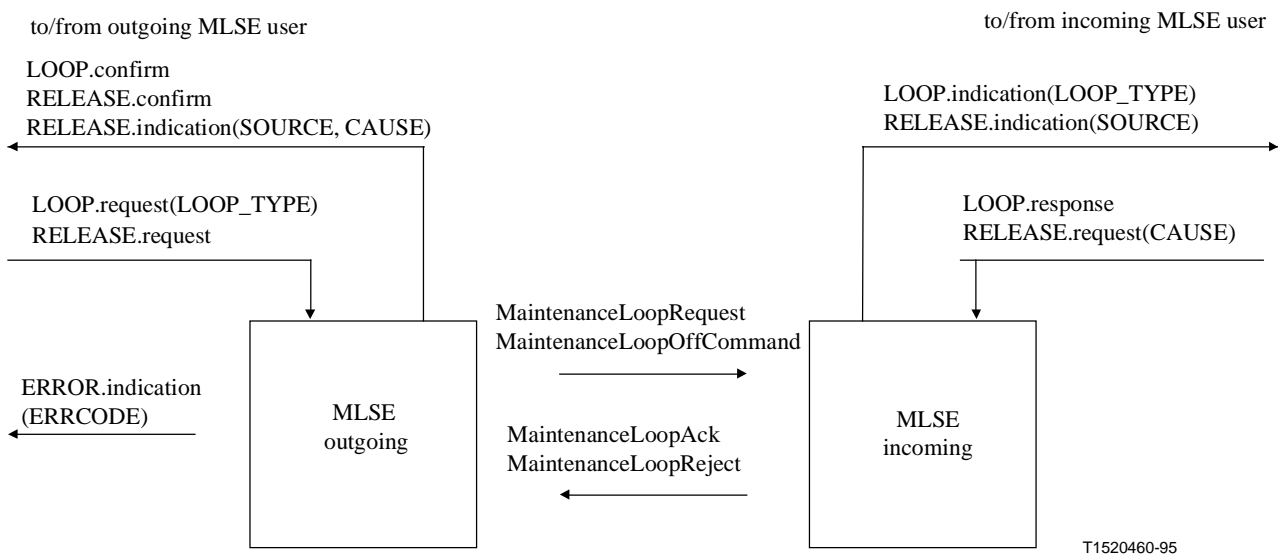


Figure C.45/H.245 – Primitives and messages in the Maintenance Loop Signalling Entity

C.11.4.2 Primitive parameter default values

Where not explicitly stated in the SDL diagrams the parameters of the indication and confirm primitives assume values as shown in Table C.43.

Table C.43/H.245 – Default primitive parameter values

Primitive	Parameter	Default value (Note)
LOOP.indication	LOOP_TYPE	MaintenanceLoopRequest.type
RELEASE.indication	SOURCE	USER
	CAUSE	MaintenanceLoopReject.cause
NOTE – A primitive parameter shall be coded as null, if an indicated message field is not present in the message.		

C.11.4.3 Message field default values

Where not explicitly stated in the SDL diagrams the message fields assume values as shown in Table C.44.

Table C.44/H.245 – Default message field values

Message	Field	Default value (Note 1)
MaintenanceLoopRequest	type	LOOP.request(LOOP_TYPE) and out_MLN (Note 2)
MaintenanceLoopAck	type	in_LOOP and in_MLN (Note 3)
MaintenanceLoopReject	type cause	in_LOOP and in_MLN (Note 3) RELEASE.request(CAUSE)
MaintenanceLoopOffCommand	–	–
NOTE 1 – A message field shall not be coded, if the corresponding primitive parameter is null i.e. not present.		
NOTE 2 – The value of the type field is derived from the LOOP_TYPE parameter and the logical channel number.		
NOTE 3 – The value of the type field is derived from the in_LOOP and in_MLN state variables.		

C.11.4.4 ERRCODE parameter values

The ERRCODE parameter of the ERROR.indication primitive indicates a particular error condition. Table C.45 shows the values that the ERRCODE parameter may take at the outgoing MLSE. There is no ERROR.indication primitive associated with the incoming MLSE.

Table C.45/H.245 – ERRCODE parameter values at outgoing MLSE

Error type	Error code	Error condition	State
Inappropriate message	A	MaintenanceLoopAck	LOOPED
No response from peer MLSE	B	Timer T102 expiry	AWAITING RESPONSE

C.11.4.5 SDLs

The outgoing MLSE and the incoming MLSE procedures are expressed in SDL form in Figure C.46 and Figure C.47 respectively.

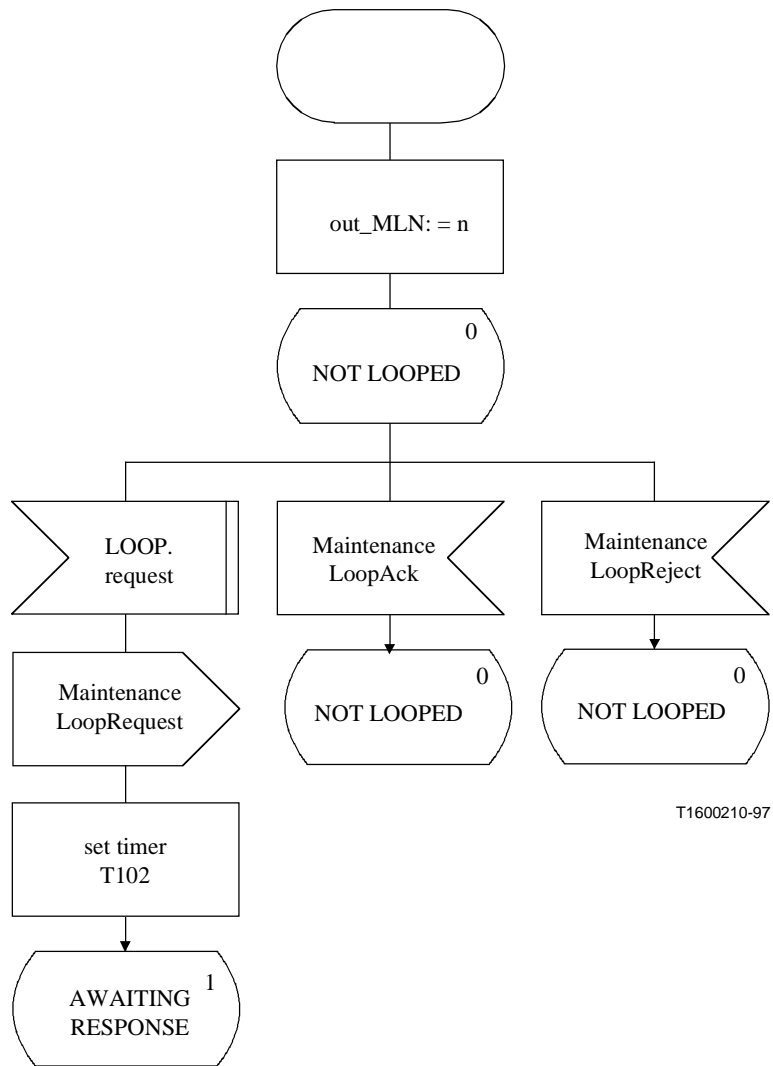


Figure C.46/H.245 – Outgoing MLSE SDL (sheet 1 of 3)

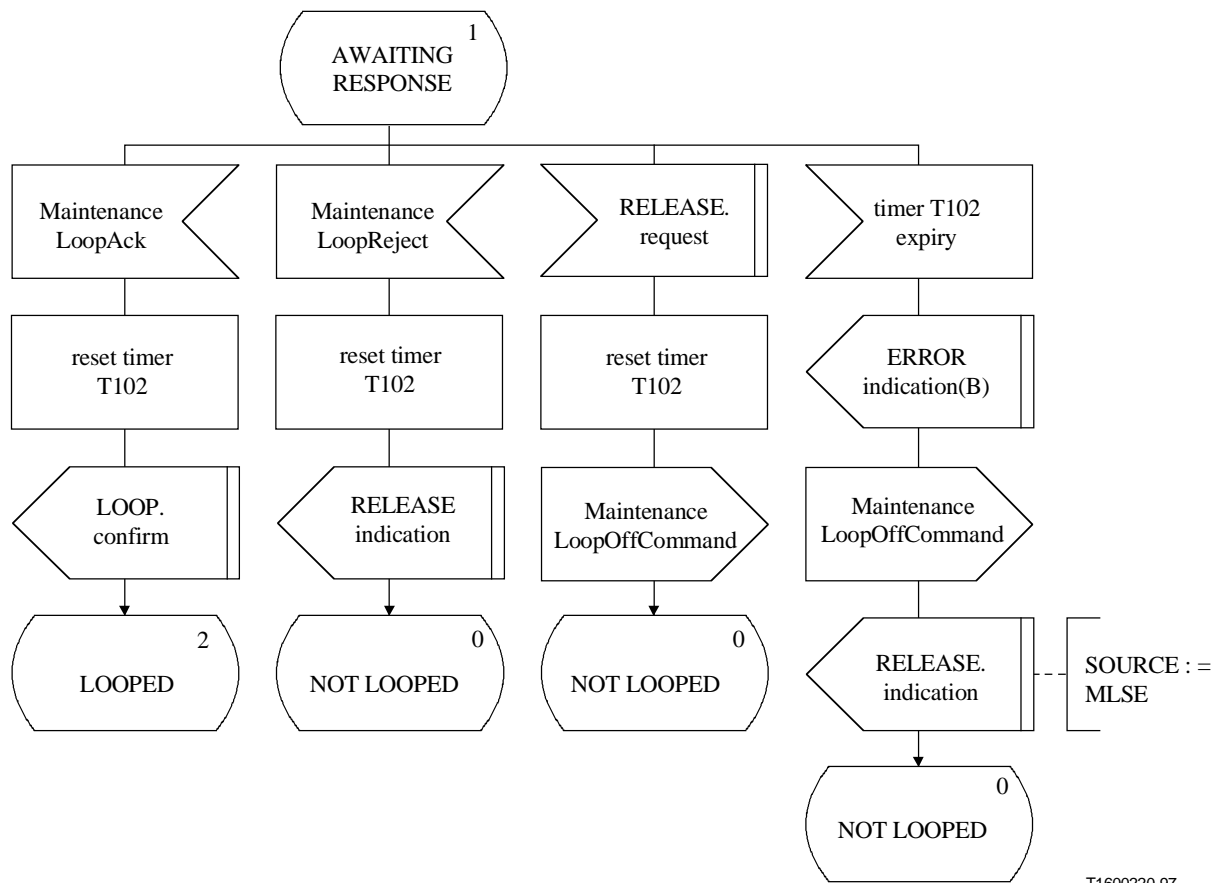


Figure C.46/H.245 – Outgoing MLSE SDL (sheet 2 of 3)

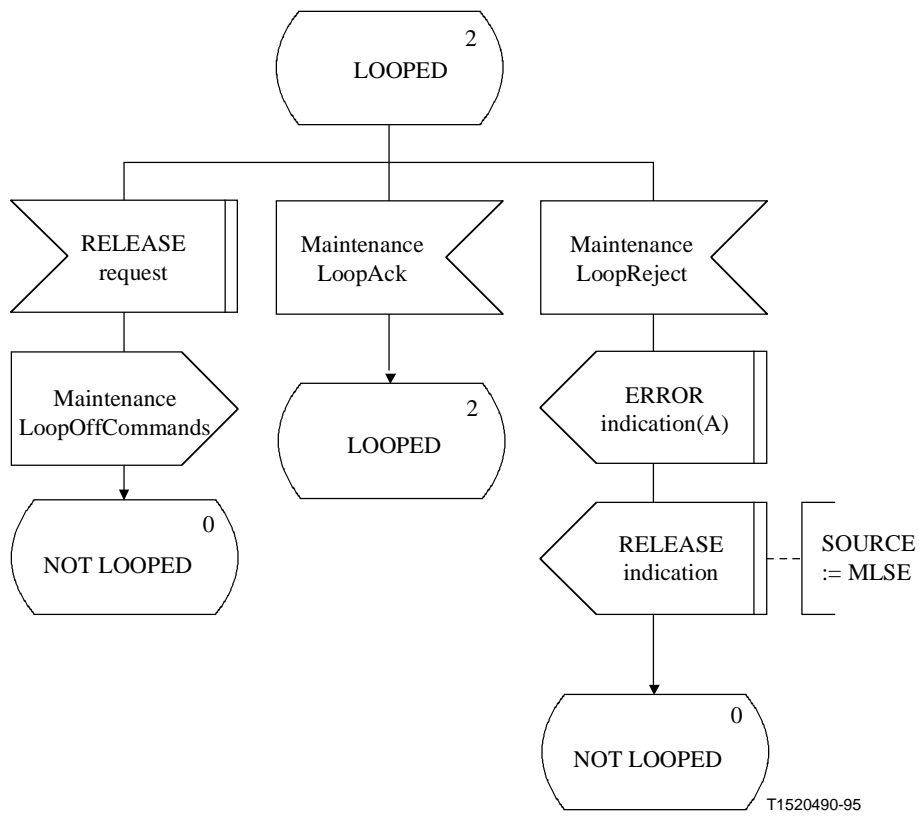


Figure C.46 /H.245 – Outgoing MLSE SDL (sheet 3 of 3)

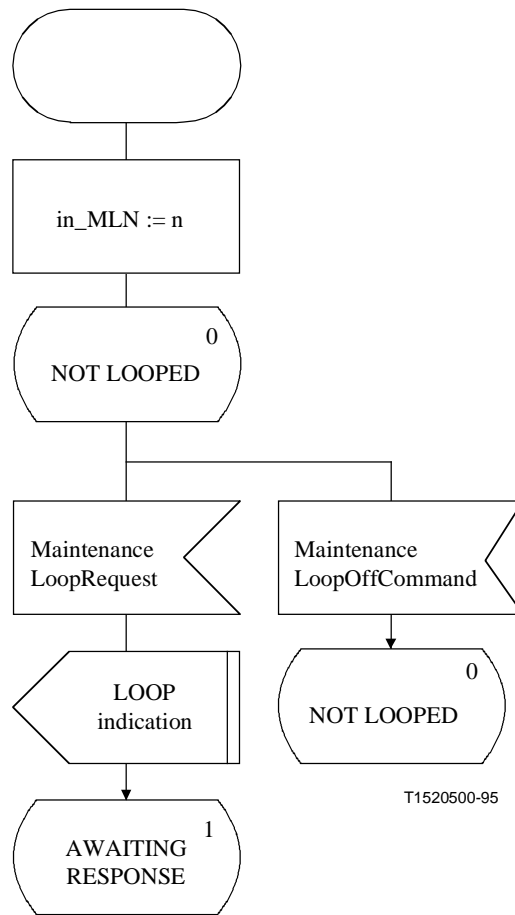


Figure C.47/H.245 – Incoming MLSE SDL (sheet 1 of 3)

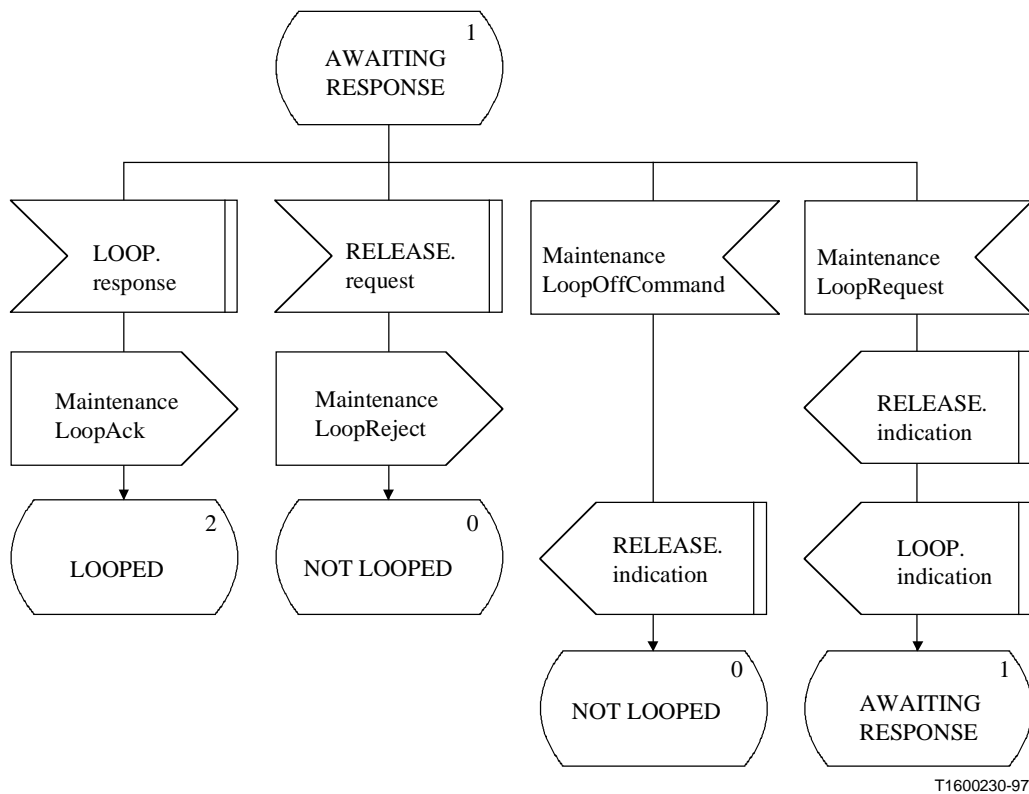


Figure C.47/H.245 – Incoming MLSE SDL (sheet 2 of 3)

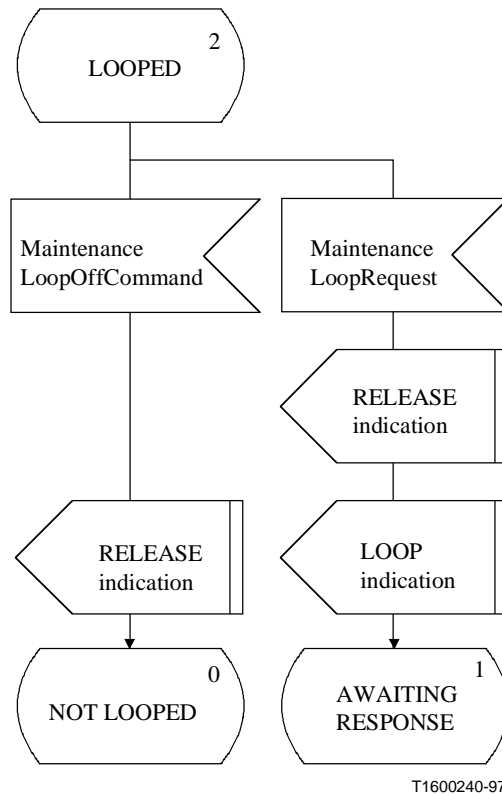


Figure C.47/H.245 – Incoming MLSE SDL (sheet 3 of 3)

ANNEX D

Object identifier assignments

Table D.1 lists the assignment of Object Identifiers defined for use by this Recommendation.

Table D.1/H.245

Object Identifier Value	Description
{itu-t (0) recommendation (0) h (8) 245 version (0) 1 }	This Object Identifier is used to indicate the version of this Recommendation in use as a multimedia system control protocol. This indicates the first version of this Recommendation.
{itu-t (0) recommendation (0) h (8) 245 version (0) 2 }	This Object Identifier is used to indicate the version of this Recommendation in use as a multimedia system control protocol. At this time there are four standardized versions defined. This indicates the second version of this Recommendation.
{itu-t (0) recommendation (0) h (8) 245 version (0) 3 }	This Object Identifier is used to indicate the version of this Recommendation in use as a multimedia system control protocol. At this time there are four standardized versions defined. This indicates the third version of this Recommendation.
{itu-t (0) recommendation (0) h (8) 245 version (0) 4 }	This Object Identifier is used to indicate the version of this Recommendation in use as a multimedia system control protocol. At this time there are four standardized versions defined. This indicates the fourth version of this Recommendation.

APPENDIX I

Overview of ASN.1 syntax

I.1 Introduction to ASN.1

Abstract Syntax Notation One (ASN.1) is a data specification language. It was originally standardized as part of the X.400 electronic mail series as ITU-T Rec. X.409. This evolved to ITU-T Rec. X.208 and most recently to ITU-T Rec. X.680. ASN.1 allows unambiguous specification of complex data structures including those with variable-length fields, optional fields and recursion.

The above-mentioned Recommendations deal only with the syntax and semantics of ASN.1 specifications. The binary encoding of data structures is covered in other Recommendations, notably ITU-T Rec. X.690 (basic encoding rules or BER) and ITU-T Rec. X.691 (packed encoding rules or PER). BER allows data to be deciphered by systems that have general knowledge of ASN.1 but do not know the details of the specification used to form the data. In other words, the data types are encoded along with the data values. PER is much more efficient since only data values are encoded

and the coding is designed with very little redundancy. This method can be used when both the transmitter and the receiver expect data to adhere to a known structure.

This Recommendation is implemented using the packed encoding rules. Since both sides of a call know that messages will conform to the H.245 specification it is not necessary to encode that specification into the messages. For decoding simplicity, the aligned variant of PER is used. This forces fields that require eight or more bits to be aligned on octet boundaries and to consume an integral number of octets. Alignment is done by padding the data with zeros before large fields.

I.2 Basic ASN.1 data types

The simplest data type is BOOLEAN, which represents the values FALSE and TRUE. These are encoded in a single bit as 0 and 1 respectively. For example, `segmentableFlag` BOOLEAN is coded:

Value	Encoding
FALSE	0
TRUE	1

The most fundamental data type is INTEGER, which represents whole number values. Integers can be unconstrained as in:

`bitRate` INTEGER

or they can be constrained to a range of values, for example:

`maximumA12SDUSize` INTEGER (0..65535)

Constrained integers are encoded differently depending on the size of the range. Suppose N is the number of integers in the range, i.e. the upper limit minus the lower limit plus one. Depending on N, the constrained integer will be encoded in one of five ways:

N	Encoding
1	No bits needed
2-255	An unaligned field of 1 to 8 bits
256	An aligned 8-bit field
257-65536	An aligned 16-bit field
Larger	As the minimum number of aligned octets preceded by the above encoding of the number of octets

In all cases, the number that is actually used is the value to be encoded minus the lower limit of the range. In these examples "pad" represents zero to seven 0 bits that are added to the encoding so that the following field will start on a 8-bit boundary.

`firstGOB` INTEGER (0..17)

Value	Encoding
0	00000
3	00011

h233IVResponseTime

INTEGER (0..255)

Value	Encoding
3	pad 00000011
254	pad 11111110

skew

INTEGER (0..4095)

Value	Encoding
3	pad 00000000 00000011
4095	pad 00001111 11111111

Unconstrained (2's complement) integer values that can be represented in 127 octets or less are encoded in the minimum number of octets needed. The number of octets (the length) is encoded as an aligned octet that precedes the number itself. For example,

```

-1          pad 00000001 11111111
0           pad 00000001 00000000
128        pad 00000010 00000000 10000000
1000000    pad 00000011 00001111 01000010 01000000

```

ASN.1 supports a variety of string data types. These are variable-length lists of bits, octets or other short data types. They are typically encoded as a length followed by the data. The length can be encoded as an unconstrained integer or as a constrained integer if the SIZE of the string is specified. For example:

data

OCTET STRING

Since the length of the octet string is not bounded, it will have to be encoded as a *semi-constrained whole number* (has a lower bound, but no upper bound). First, the data is padded so that the encoding will be aligned. The rest of the code is as follows:

Length	Encoding
0 to 127	8-bit length followed by the data
128 to 16K-1	16-bit length with the MSB set, then the data
16K to 32K-1	11000001, 16K octets of data, then code the rest
32K to 48K-1	11000010, 32K octets of data, then code the rest
48K to 64K-1	11000011, 48K octets of data, then code the rest
64K or more	11000100, 64K octets of data, then code the rest

This method is called "fragmentation". Note that if the length is a multiple of 16K, then the representation will end with an octet of zero indicating a zero-length string.

I.3 Aggregate data types

ASN.1 includes several aggregate or container data types that are similar in concept to C's union, struct and array types. These are, respectively, CHOICE, SEQUENCE and SEQUENCE OF. In all cases they are encoded with some bits specific to the container followed by the normal encodings of the contents.

CHOICE is used to select exactly one of a group of data types. For example:

```

VideoCapability ::= CHOICE
{
    nonStandardNonStandardParameter,
    h261VideoCapability           H261VideoCapability,
    h262VideoCapability         H262VideoCapability,
    h263VideoCapability         H263VideoCapability,
    is11172VideoCapability      IS11172VideoCapability,
    ...
}

```

An index number is assigned to each choice, starting with zero. The index of the actual choice is encoded as a constrained integer. The index is followed by the encoding of the actual selection or nothing if the selection is **NULL**. If the extension marker is present (as above), the index is preceded by a bit that is zero if the actual choice is from the original list.

SEQUENCE is simply a grouping of dissimilar data types. Individual elements of the sequence may be **OPTIONAL**. The encoding is very simple. If there is an extension marker the first bit indicates the presence of additional elements. This is followed by a series of bits, one for each optional element that indicates if that data is present. This is followed by the encodings of the components of the sequence. For example:

```

H261VideoCapability ::= SEQUENCE
{
    qcifMPI           INTEGER (1..4) OPTIONAL,           -- units 1/29.97 Hz
    cifMPI            INTEGER (1..4) OPTIONAL,           -- units 1/29.97 Hz
    temporalSpatialTradeOffCapability BOOLEAN,
    ...
}

```

The encoding has one bit for the extension marker, two bits for the optional fields, two bits each for any optional field that is present, one bit for the boolean and then any extension data. Note that in this sequence has no padding for octet alignment.

The **SEQUENCE OF** and **SET OF** types describe a collection of similar components (an array). **SEQUENCE OF** implies that the order of the elements is significant, with **SET OF** the element order is arbitrary. The **PER** encoding is the same for both types.

These types can have a **SIZE** constraint or an unconstrained number of elements. If the number is known *a priori* and is less than 64K, it is not encoded. Otherwise the actual number of components is encoded as a constrained or semi-constrained length. This is followed by the encoding of the data. If the length is at least 16K and is encoded then the list of data will be broken into fragments like the octet string. In this case the fragments are broken after some number of component fields (16K, 32K, etc.), not after some number of octets.

I.4 Object Identifier type

Normally the type of a value is given in the ASN.1 specification so that the only information that needs to be coded and transmitted is the data itself. Occasionally, however, it is desirable to encode the data type as well as the data value. For example, **protocolIdentifier** contains:

```

protocolIdentifier OBJECT IDENTIFIER,
    -- shall be set to the value
    -- {itu-t (0) recommendation (0) h (8) 245 version (0) 1}

```

This is encoded as the data encoded with the BER (ITU-T Rec. X.690) preceded by the length of that encoding in octets. The length is encoded as a semi-constrained whole number (see the OCTET STRING example above). The following illustrates how this is encoded.

The first octet indicates the length of the encoding that follows.

The first two components of the object identifier are combined together as $40 \times \text{first one} + \text{second one}$, in this case $40 \times 0 + 0 = 0$. The others are encoded as they are. Each is encoded into a series of octets, the first bit of which indicates whether there is any more. So:

$0 \rightarrow 0000\ 0000$,

$8 \rightarrow 0000\ 0100$,

while 245, being more than 127 becomes $1000\ 0001\ 0111\ 0101$.

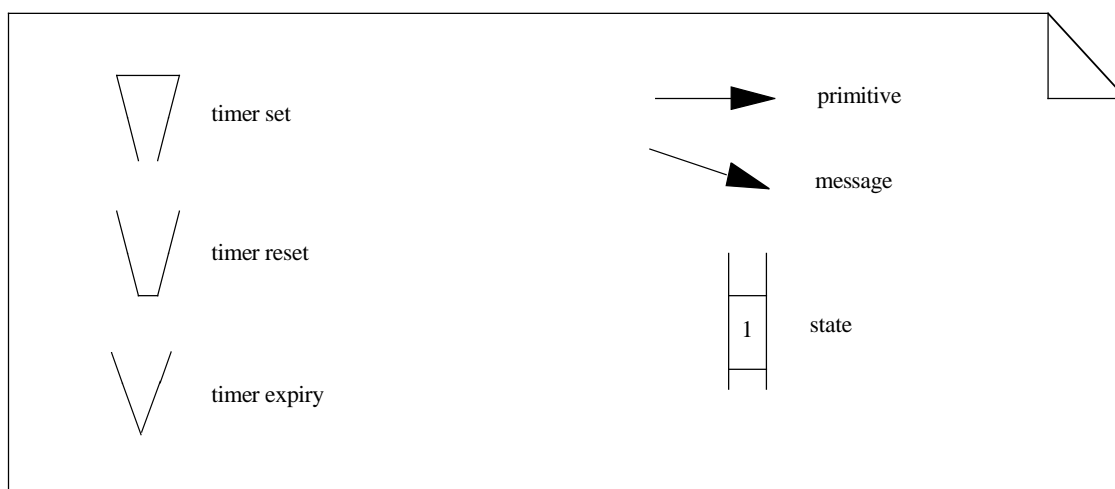
So the entire encoding in hexadecimal consists of the seven octets 06 00 08 81 75 00 01.

APPENDIX II

Examples of H.245 procedures

II.1 Introduction

This appendix illustrates examples of the procedures defined in clause C.8. Figure II.1-1 shows the key to diagrams used in this appendix.



T1523580-96

Figure II.1-1/H.245 – Key to figures

II.2 Master Slave Determination Signalling Entity

In the following figures messages are represented by the shortened names given in Table II.2-1.

Table II.2-1/H.245 – Master slave determination shortened names

Message	Name in examples
MasterSlaveDetermination	MSD
MasterSlaveDeterminationAck	MSDAck
MasterSlaveDeterminationReject	MSDReject
MasterSlaveDeterminationRelease	MSDRelease

In the following figures IDLE, OUTGOING AWAITING RESPONSE, and INCOMING AWAITING RESPONSE states are labelled as "0", "1", and "2" respectively.

In the following figures the parameter value associated with the DETERMINE.indication and DETERMINE.confirm primitives is that of the TYPE parameter. The field value associated with the MasterSlaveDeterminationAck message is that of the decision field.

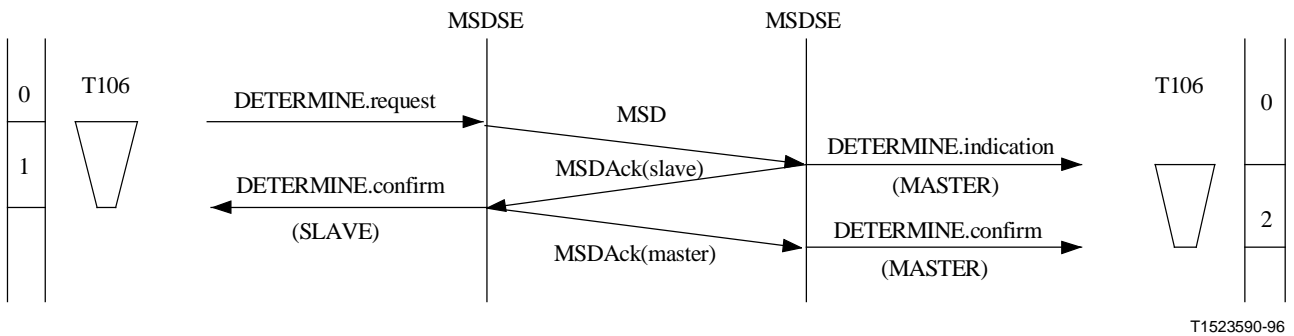


Figure II.2-1/H.245 – Master slave determination – Master at remote MSDSE

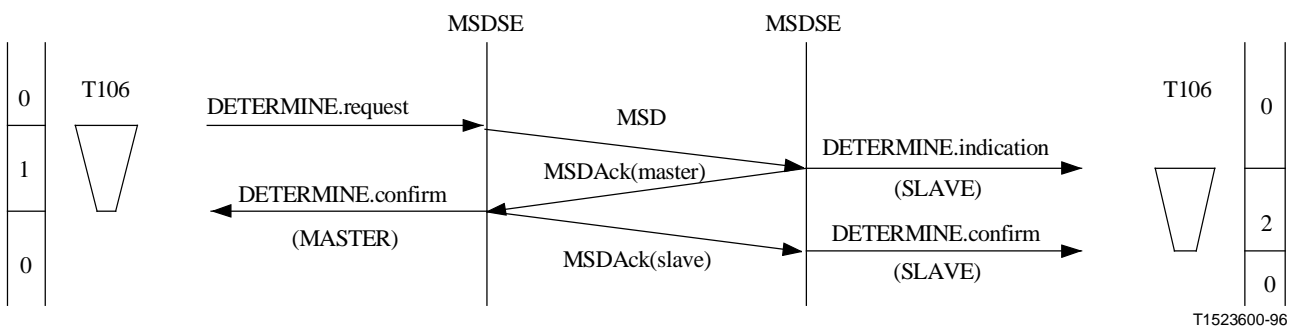


Figure II.2-2/H.245 – Master slave determination – Slave at remote MSDSE

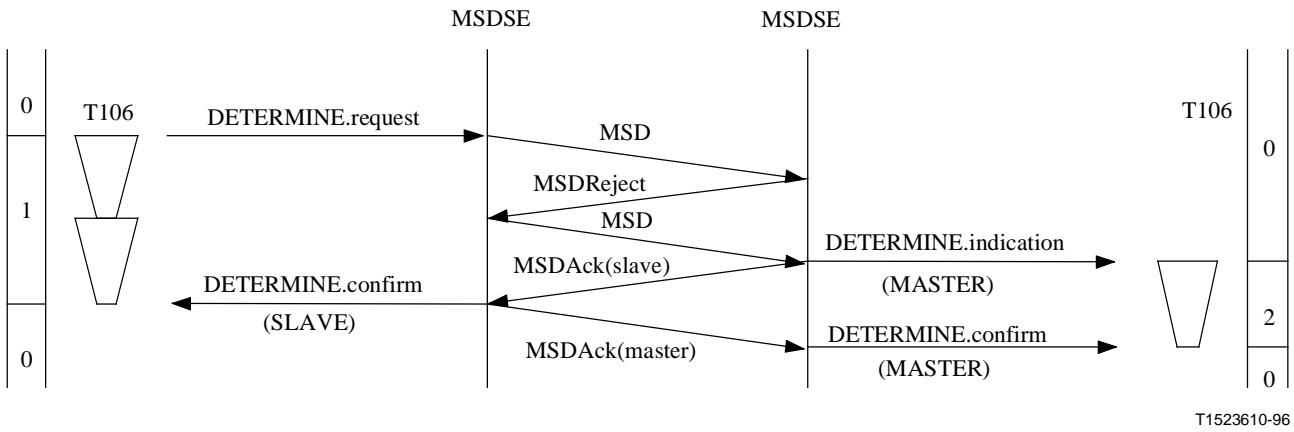


Figure II.2-3/H.245 – Master slave determination – First attempt produced an indeterminate result. The second attempt was successful

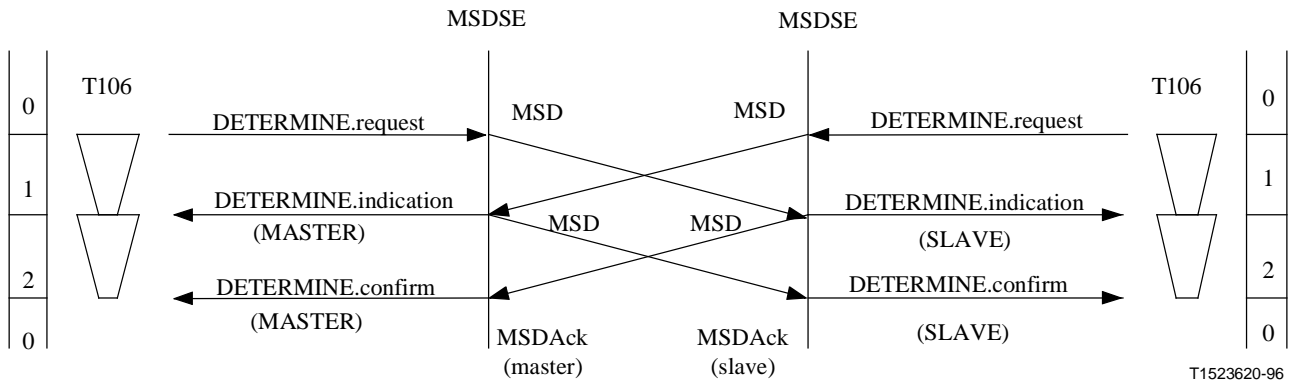


Figure II.2-4/H.245 – Master slave determination – Simultaneous determination

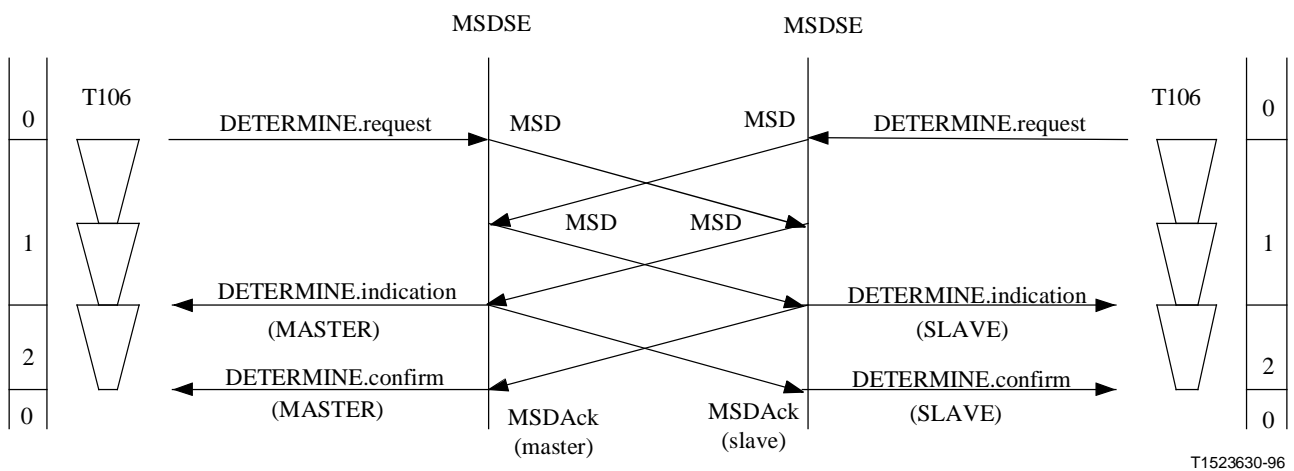


Figure II.2-5/H.245 – Master slave determination – Simultaneous determination but with the first attempt returning an indeterminate result

In Figure II.2-6 local timer T106 has expired. Only the terminal on the right knows its status. The terminal on the right is able to receive new commands but may not request anything of the other terminal that relies on knowledge of the status determination result. The terminal on the left can neither accept nor initiate new procedures. A second status determination procedure should be initiated.

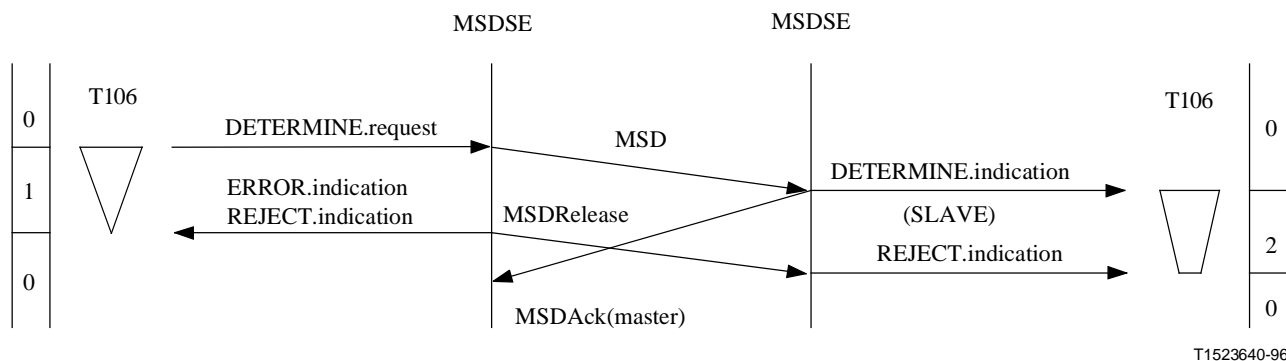


Figure II.2-6/H.245 – Master slave determination – Local timer T106 expiry with slave at remote end

In Figure II.2-7 remote timer T106 has expired during the INCOMING AWAITING ACKNOWLEDGEMENT state. Both terminals know their status. The terminal on the left may receive and issue commands. However the remote terminal does not know if the local terminal is ready to receive, and cannot issue commands that rely on knowledge of the status determination result. A second status determination procedure should be initiated.

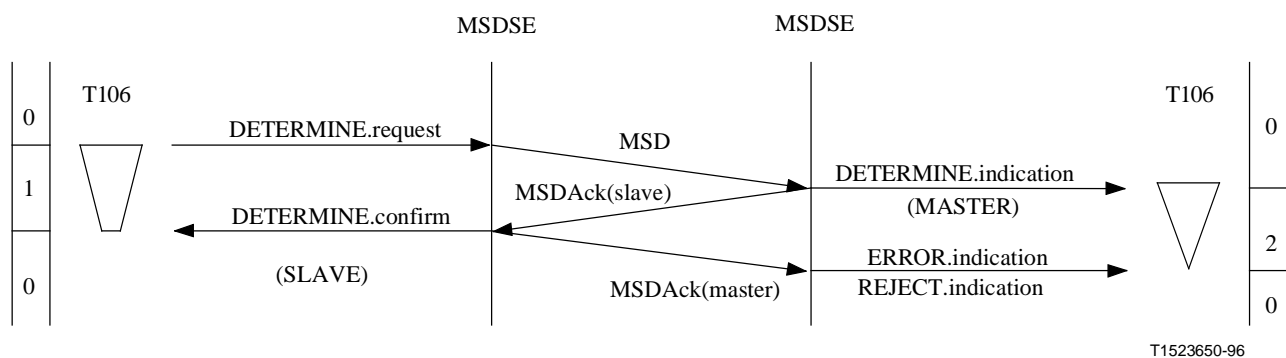
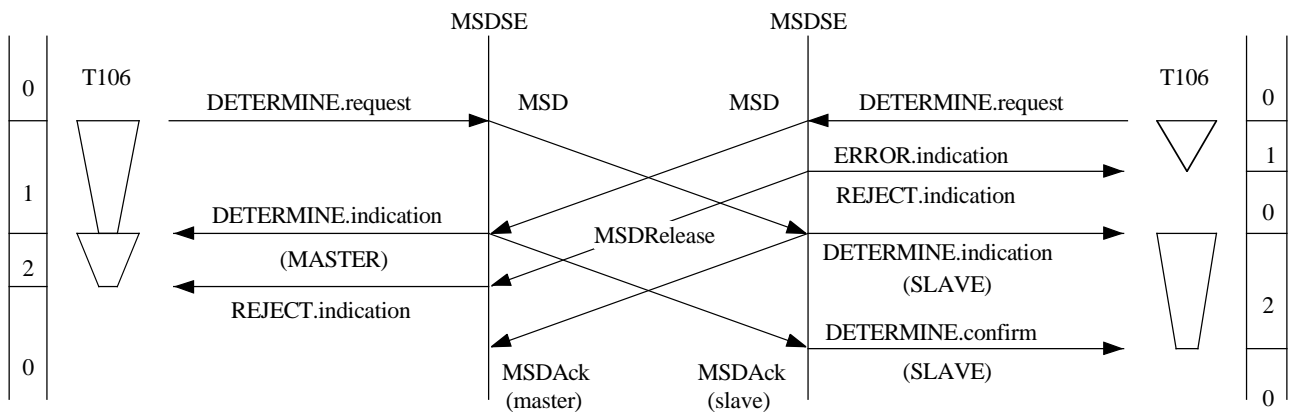


Figure II.2-7/H.245 – Master slave determination – Remote timer T106 expiry with master at remote end

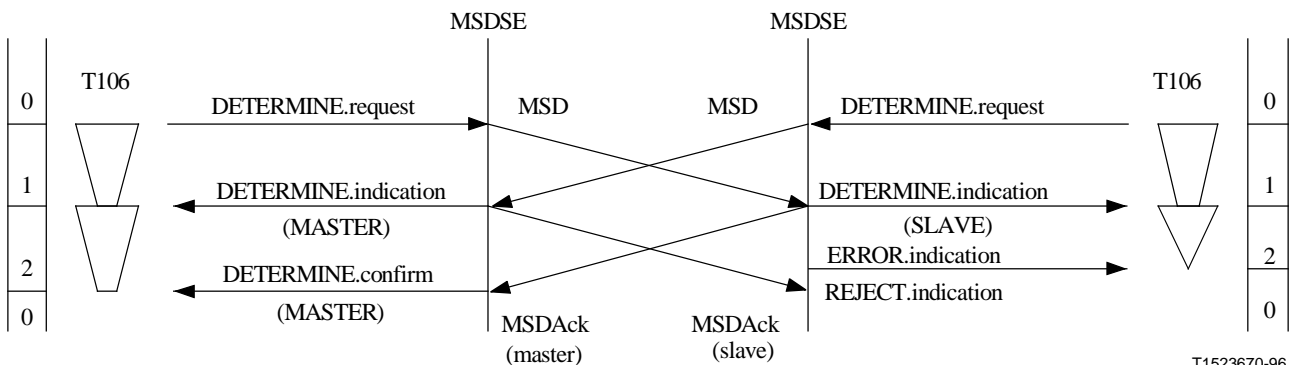
In Figure II.2-8 remote timer T106 has expired during the OUTGOING AWAITING ACKNOWLEDGEMENT state during a simultaneous determination procedure. Both terminals know their status. The terminal on the right can receive and issue commands. However the terminal on the left does not know if the other terminal is ready to receive, and cannot issue commands that rely on knowledge of the status determination result. It may receive such commands. A second status determination procedure should be initiated.



T1523660-96

Figure II.2-8/H.245 – Master slave determination – Simultaneous determination procedures with timer T106 expiry at slave

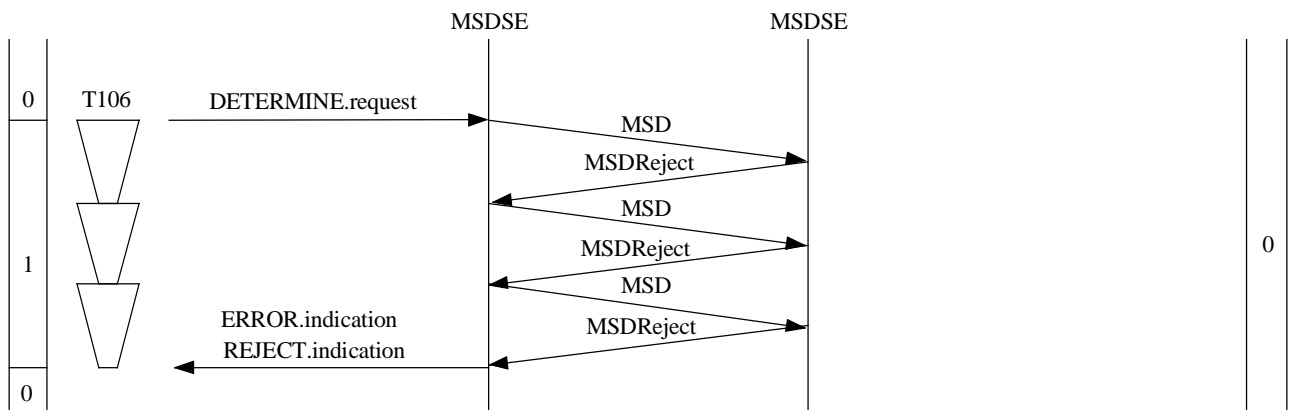
In Figure II.2-9 remote timer T106 has expired during the INCOMING AWAITING ACKNOWLEDGEMENT state, during a simultaneous determination procedure. Both terminals know their status. The terminal on the left can receive and issue commands. However the terminal on the right does not know if the other terminal is ready to receive, and cannot issue commands that rely on knowledge of the status determination result. It may receive such commands. A second status determination procedure should be initiated.



T1523670-96

Figure II.2-9/H.245 – Master slave determination – Simultaneous determination procedures with timer T106 expiry during INCOMING AWAITING ACKNOWLEDGEMENT

In Figure II.2-10 an indeterminate result was obtained N100 times. In this case N100 = 3.

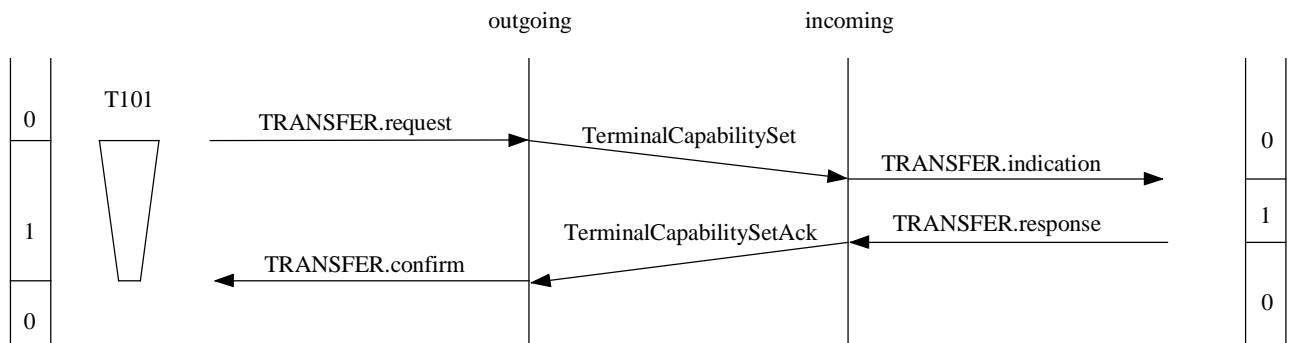


T1523680-96

Figure II.2-10/H.245 – Master slave determination – Indeterminate result with N100 = 3

II.3 Capability Exchange Signalling Entity

The following figures illustrate CESE procedures. The IDLE and AWAITING RESPONSE states are labelled as "0" and "1" respectively.



T1523690-96

Figure II.3-1/H.245 – Capability exchange with acceptance from the peer incoming CESE user

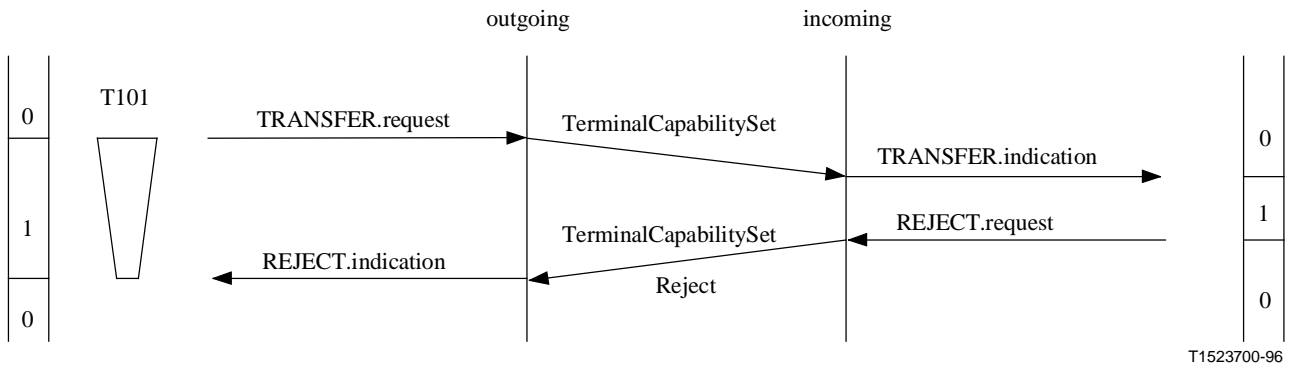
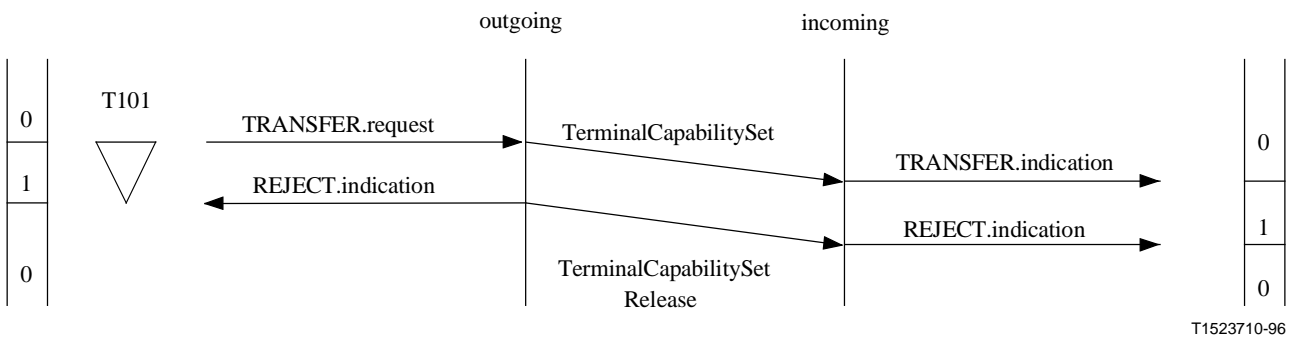
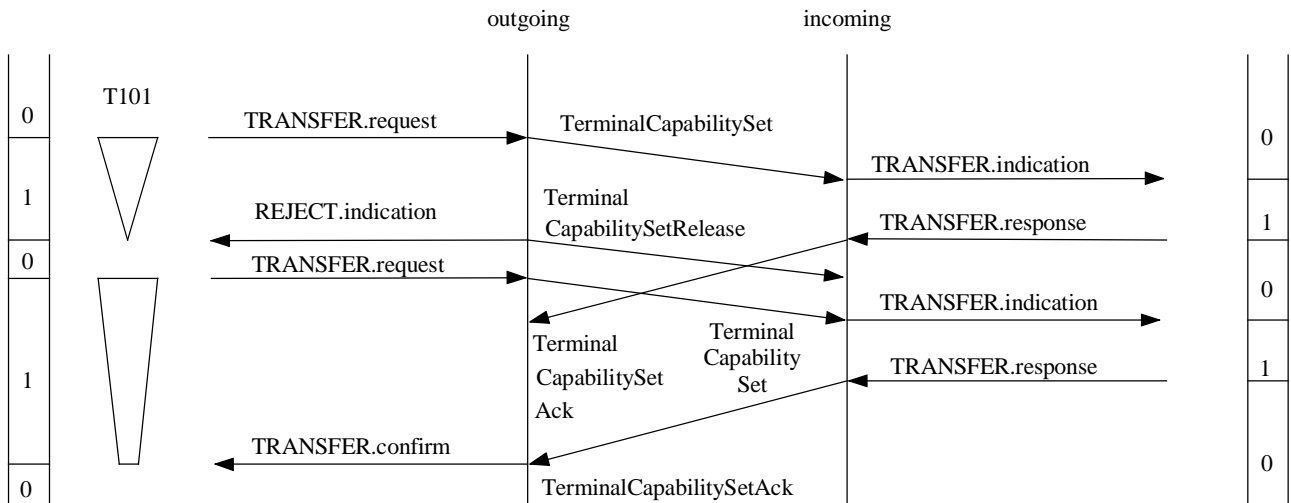


Figure II.3-2/H.245 – Capability exchange with rejection from peer incoming CESE user



NOTE – The TerminalCapabilitySetRelease message arrives at the incoming CESE before response from the incoming CESE user.

Figure II.3-3/H.245 – Capability exchange with timer T101 expiry.



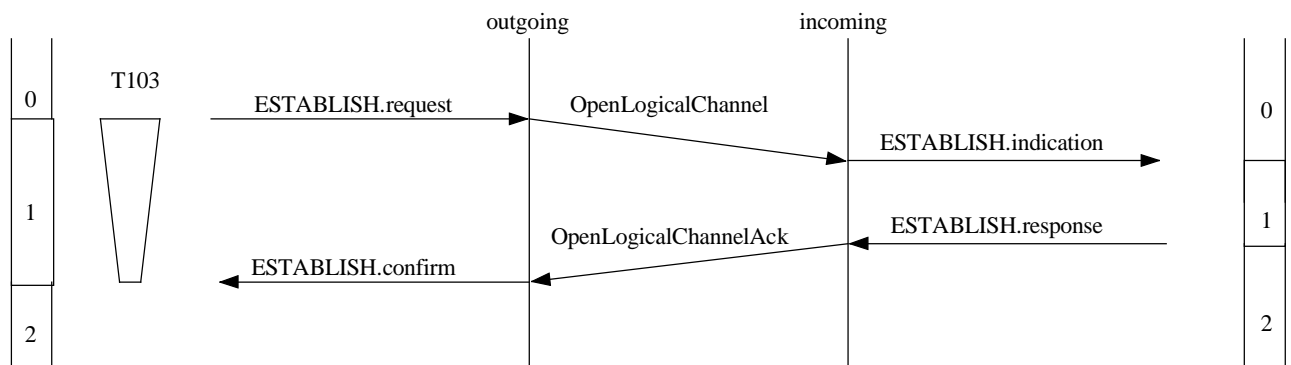
T1523720-96

NOTE – The TerminalCapabilitySetRelease message arrives at the incoming CESE after response from the incoming CESE user. At the outgoing CESE the TerminalCapabilitySetAck message in response to the first TerminalCapabilitySet message is ignored. Only the second capability exchange is successful.

Figure II.3-4/H.245 – Capability exchange with timer T101 expiry followed by a second capability exchange

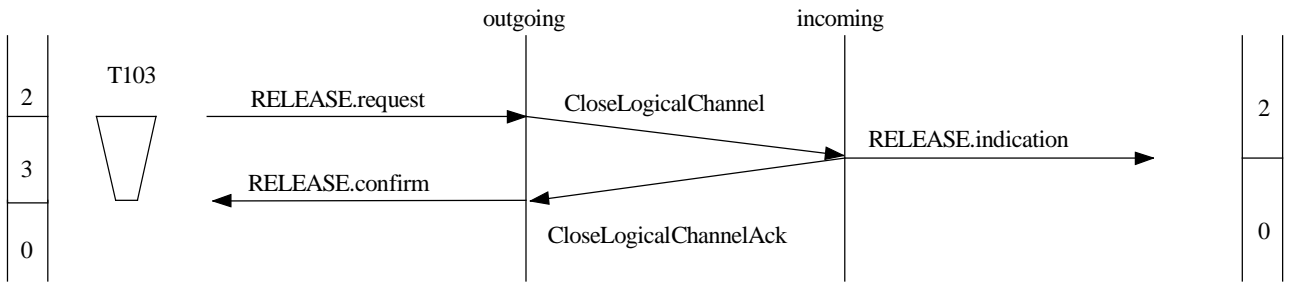
II.4 Logical Channel Signalling Entity

The following figures illustrate LCSE procedures. The outgoing LCSE states of RELEASED, AWAITING ESTABLISHMENT, ESTABLISHED, and AWAITING RELEASE are labelled as "0", "1", "2", and "3" respectively. The incoming LCSE states of RELEASED, AWAITING ESTABLISHMENT, and ESTABLISHED, are labelled as "0", "1", and "2" respectively.



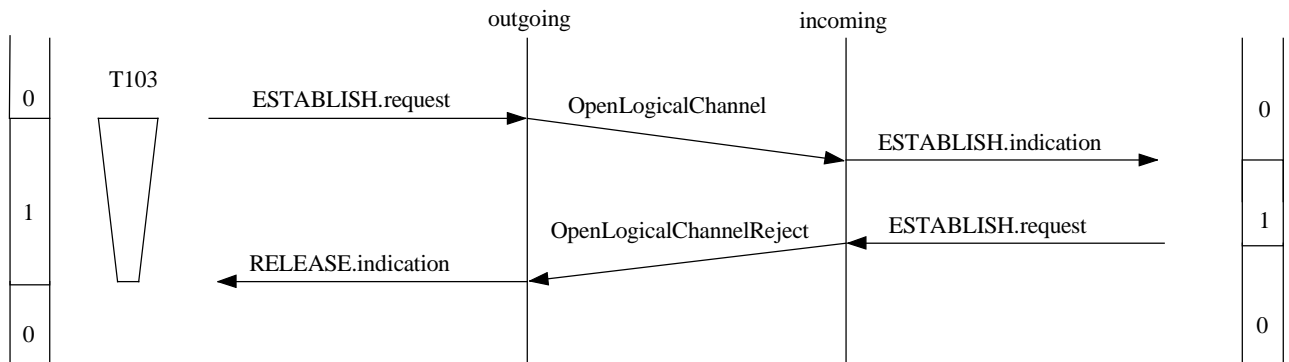
T1523730-96

Figure II.4-1/H.245 – Logical channel establishment



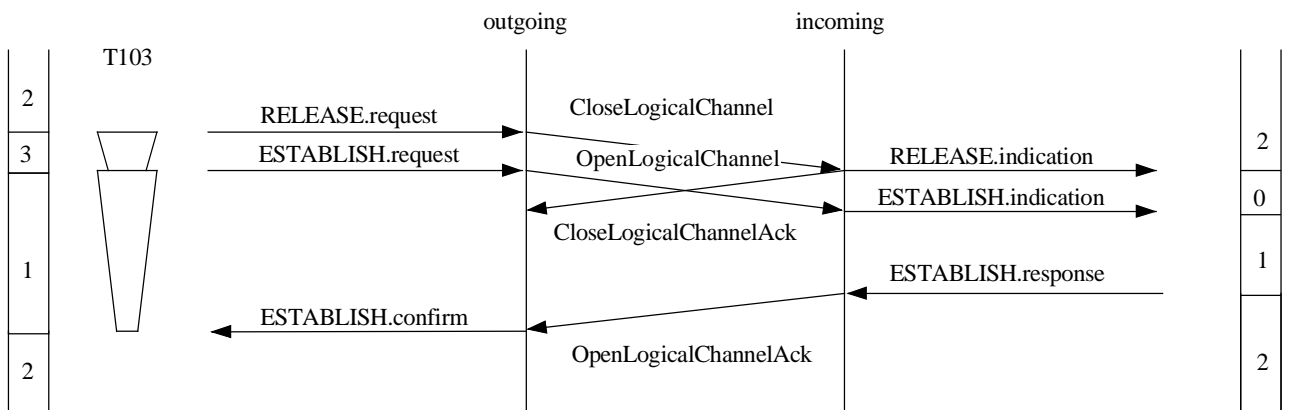
T1523740-96

Figure II.4-2/H.245 – Logical channel release



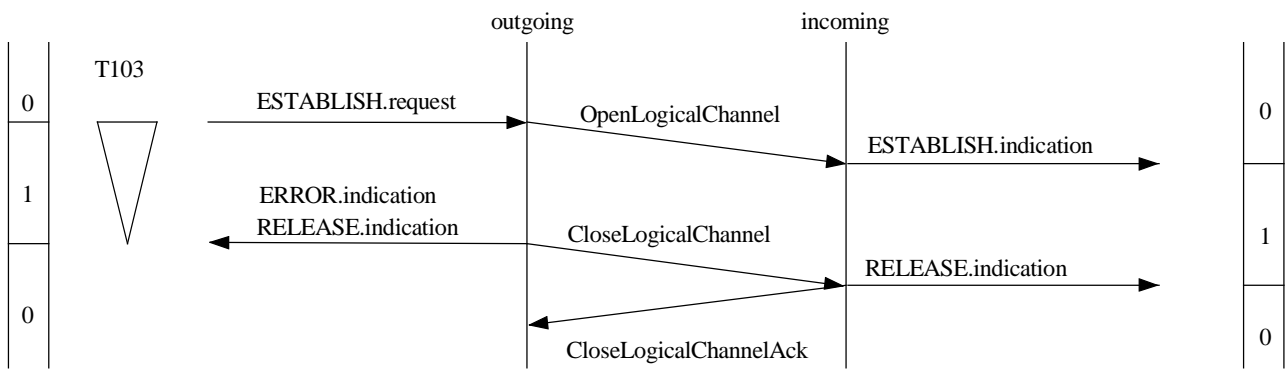
T1523750-96

Figure II.4-3/H.245 – Logical channel establishment rejection by peer LCSE user



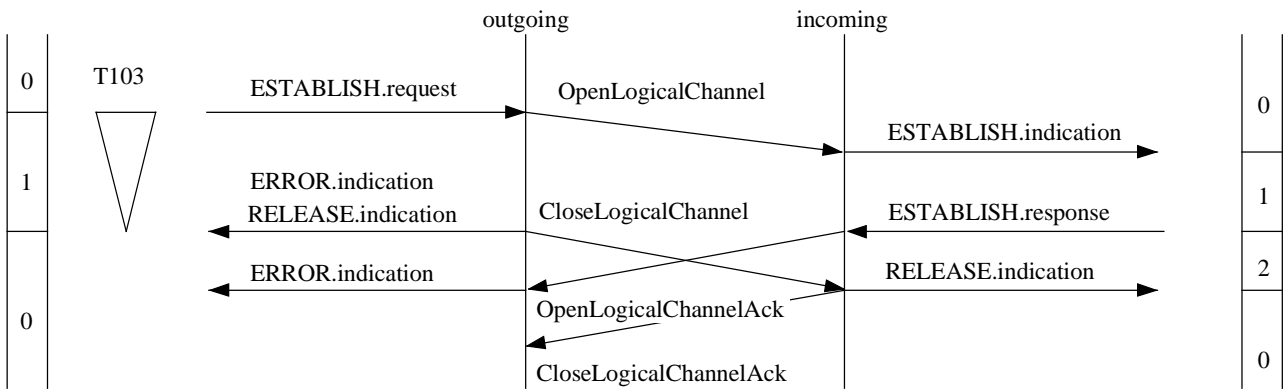
T1523760-96

Figure II.4-4/H.245 – Logical channel release followed by immediate re-establishment



T1523770-96

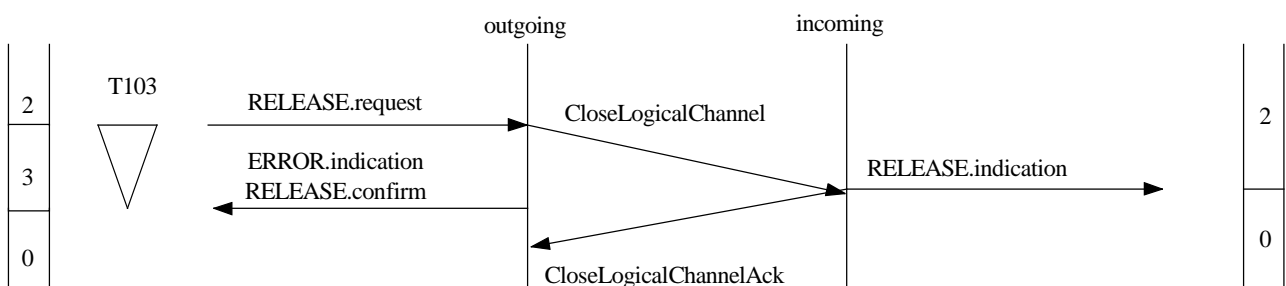
Figure II.4-5/H.245 – Logical channel establishment request with expiry of timer T103 due to slow response from peer incoming LCSE user



T1523780-96

NOTE – Timer T103 has expired after transmission of the OpenLogicalChannelAck message at the incoming LCSE, but before reception of the OpenLogicalChannelAck message at the outgoing LCSE.

Figure II.4-6/H.245 – Logical channel establishment request with expiry of timer T103



T1523790-96

Figure II.4-7/H.245 – Logical channel release request with expiry of timer T103

II.5 Close Logical Channel Signalling Entity

The following figures illustrate CLCSE procedures. The IDLE and AWAITING RESPONSE states are labelled as "0" and "1" respectively.

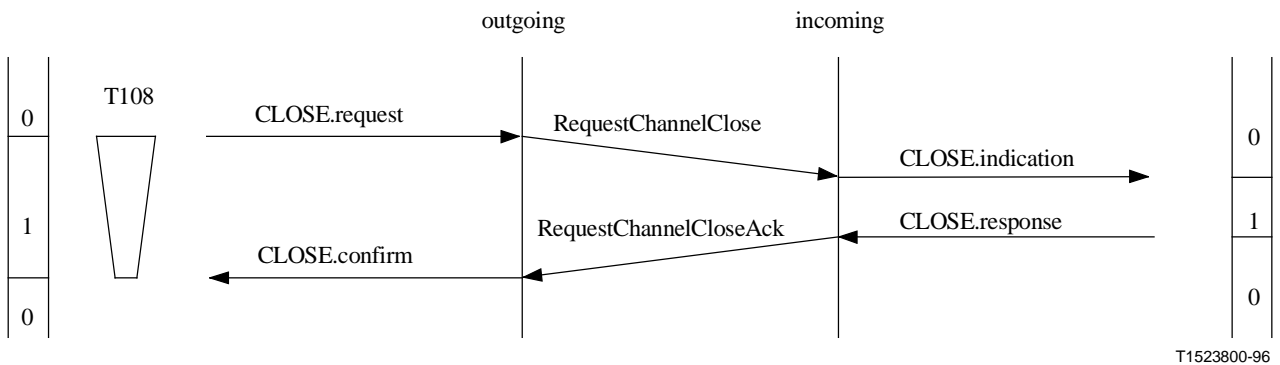


Figure II.5-1/H.245 – Close logical channel request

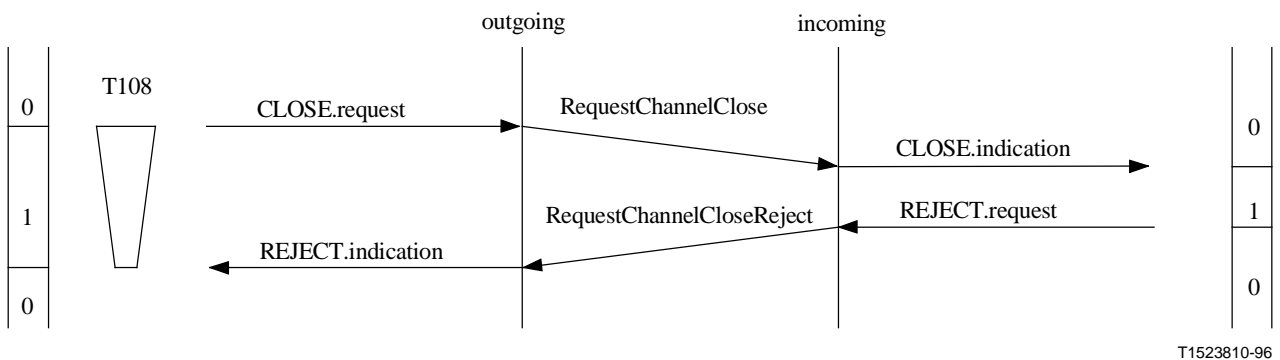
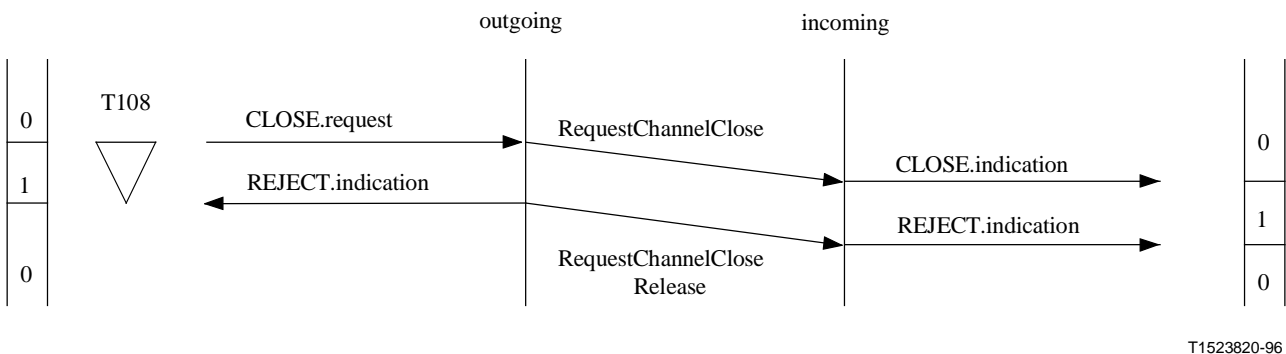
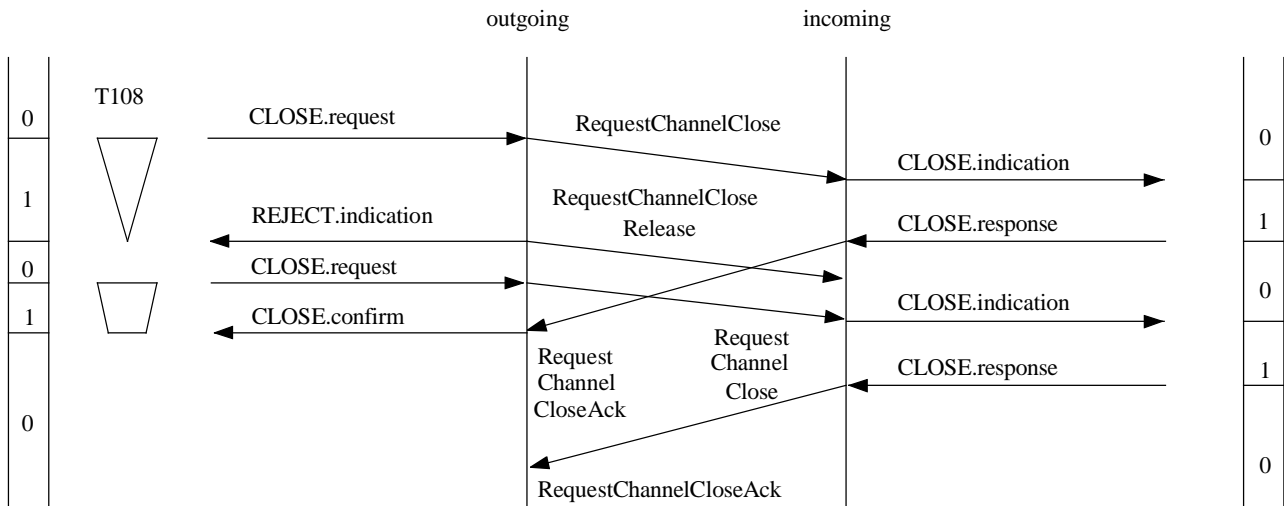


Figure II.5-2/H.245 – Close logical channel request with rejection from peer incoming CLCSE user



NOTE – The RequestChannelCloseRelease message arrives at the incoming CLCSE before response from the incoming CLCSE user.

Figure II.5-3/H.245 – Close logical channel request with timer T108 expiry



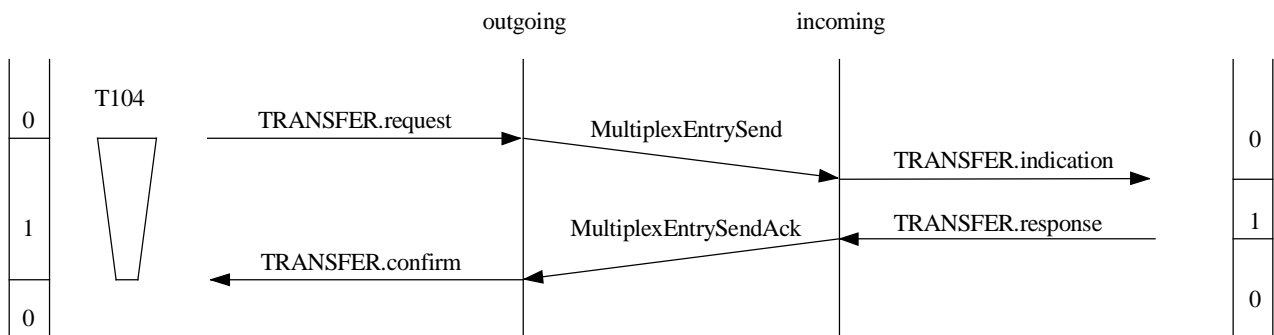
T1523830-96

NOTE – The close channel request is confirmed on reception of the first RequestChannelClose message.

Figure II.5-4/H.245 – Close logical channel request with timer T108 expiry followed by a second close logical channel request

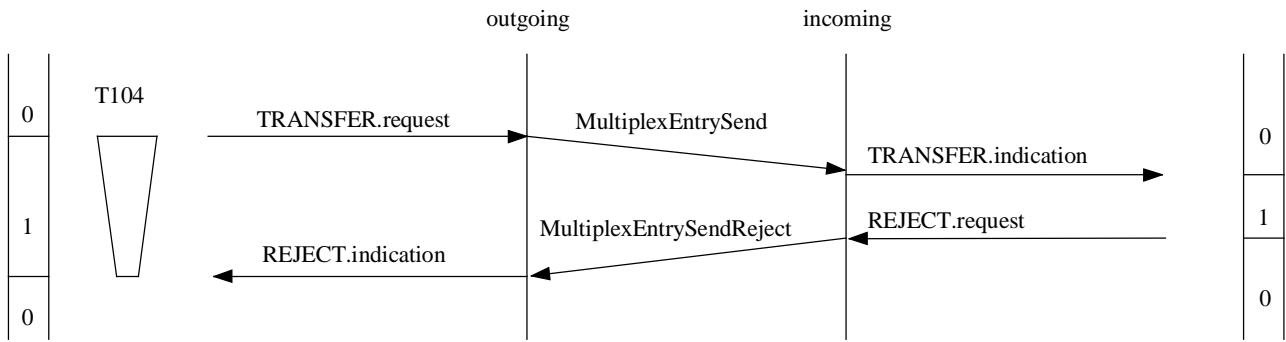
II.6 Multiplex Table Signalling Entity

The following figures illustrate MTSE procedures. The IDLE and AWAITING RESPONSE states are labelled as "0" and "1" respectively.



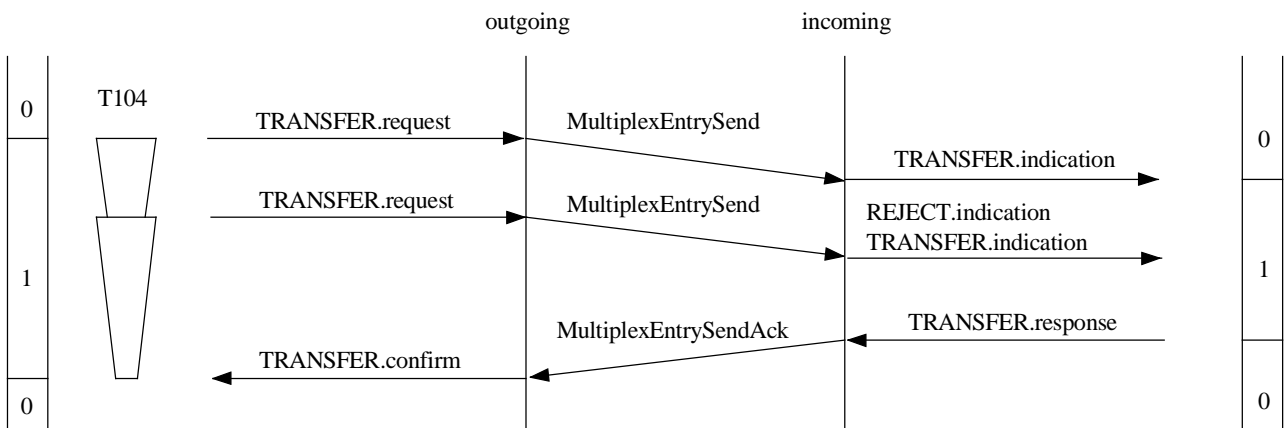
T1523840-96

Figure II.6-1/H.245 – Successful multiplex table send request



T1523850-96

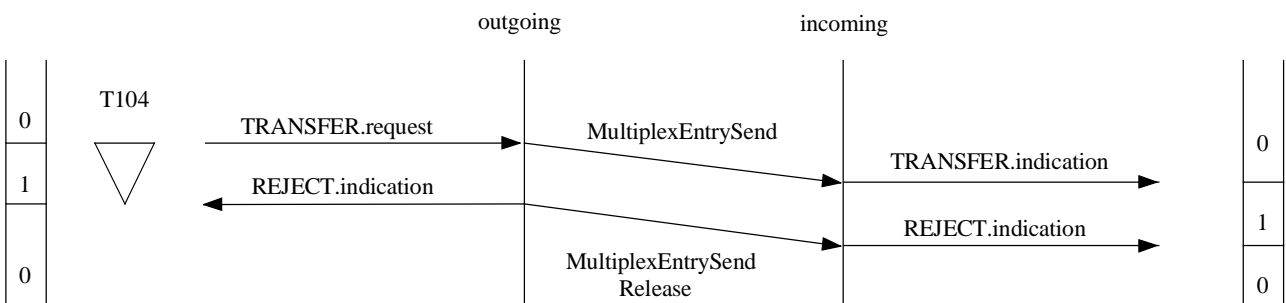
Figure II.6-2/H.245 – Multiplex table send request with rejection from the peer MTSE user



T1523860-96

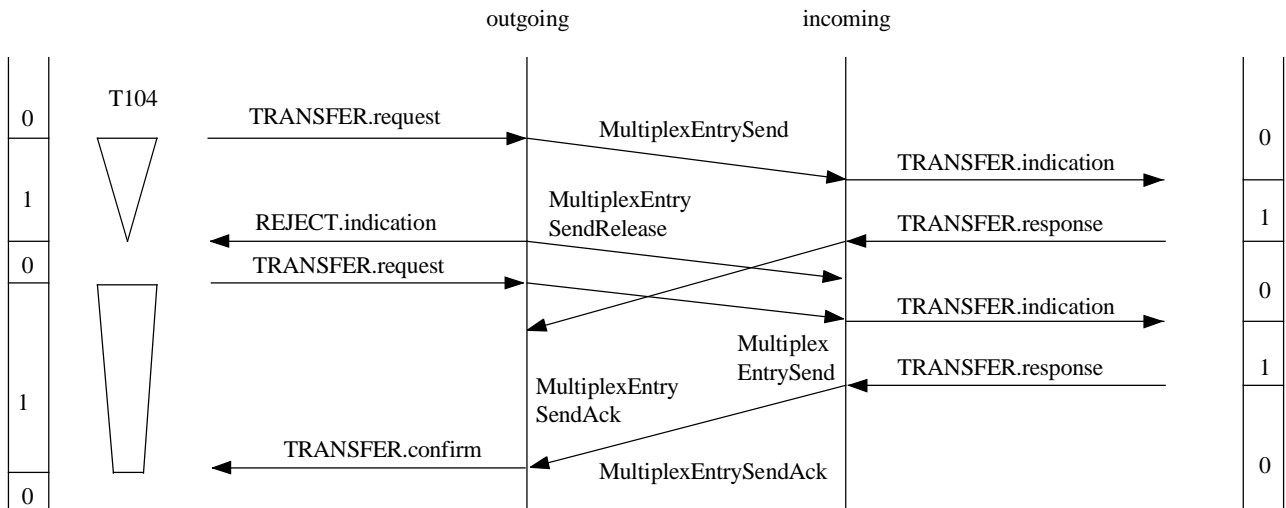
NOTE – The first request was unsuccessful.

Figure II.6-3/H.245 – Multiplex table send request with a second multiplex table send request before acknowledgement of the first request



T1523870-96

Figure II.6-4/H.245 – Multiplex table send request with timer T104 expiry



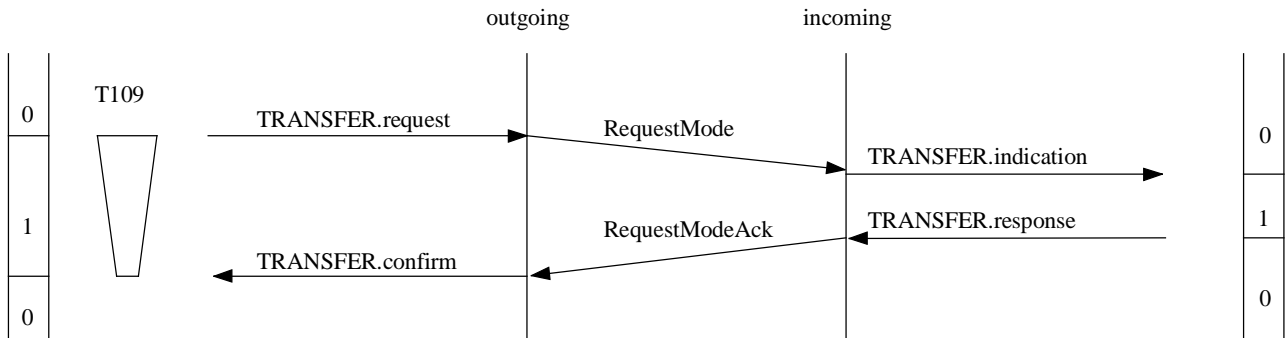
T1523880-96

NOTE – The first MultiplexEntrySendAck message is ignored at the outgoing MTSE. Only the second request was successful.

Figure II.6-5/H.245 – Multiplex table send request with timer T104 expiry followed by a second multiplex table send request

II.7 Mode Request Signalling Entity

The following figures illustrate MRSE exchanges. The IDLE and AWAITING RESPONSE states are labelled as "0" and "1" respectively.



T1523890-96

Figure II.7-1/H.245 – Successful mode request

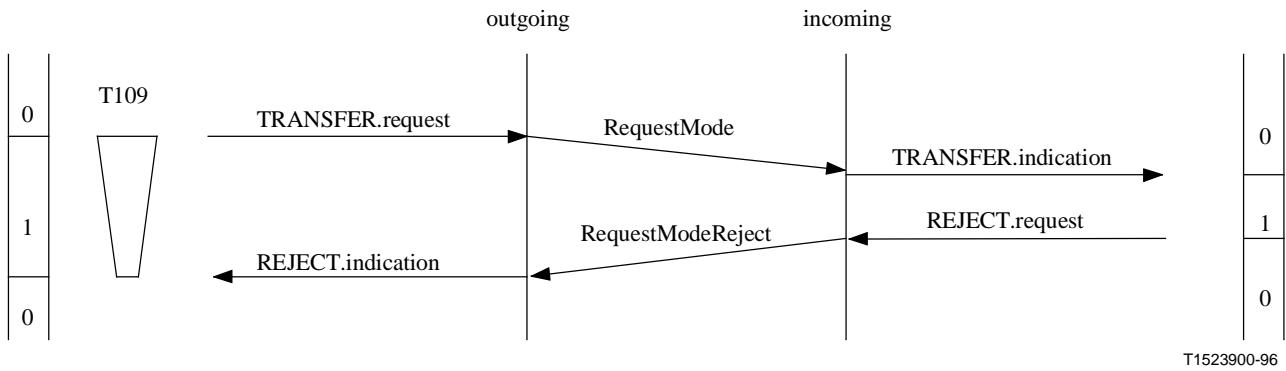
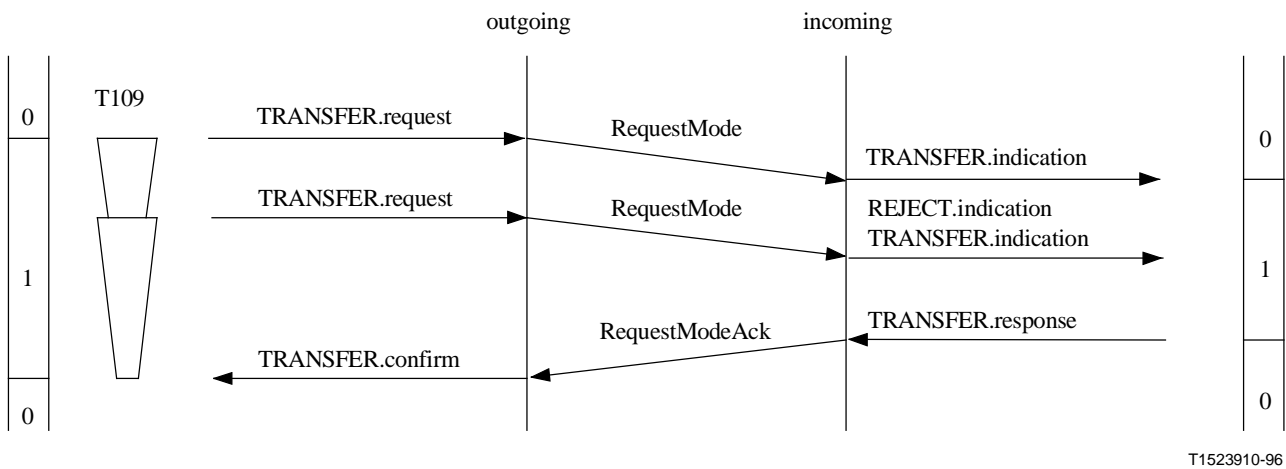
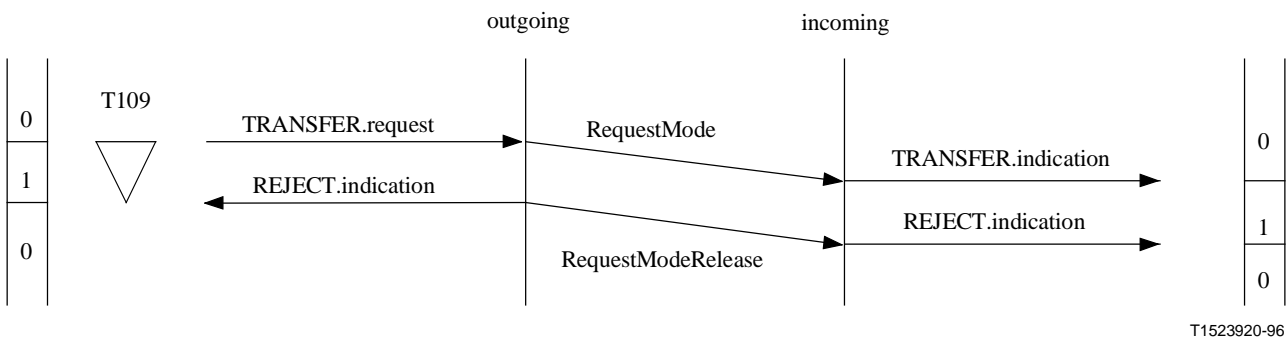


Figure II.7-2/H.245 – Mode request with rejection from the peer MRSE user



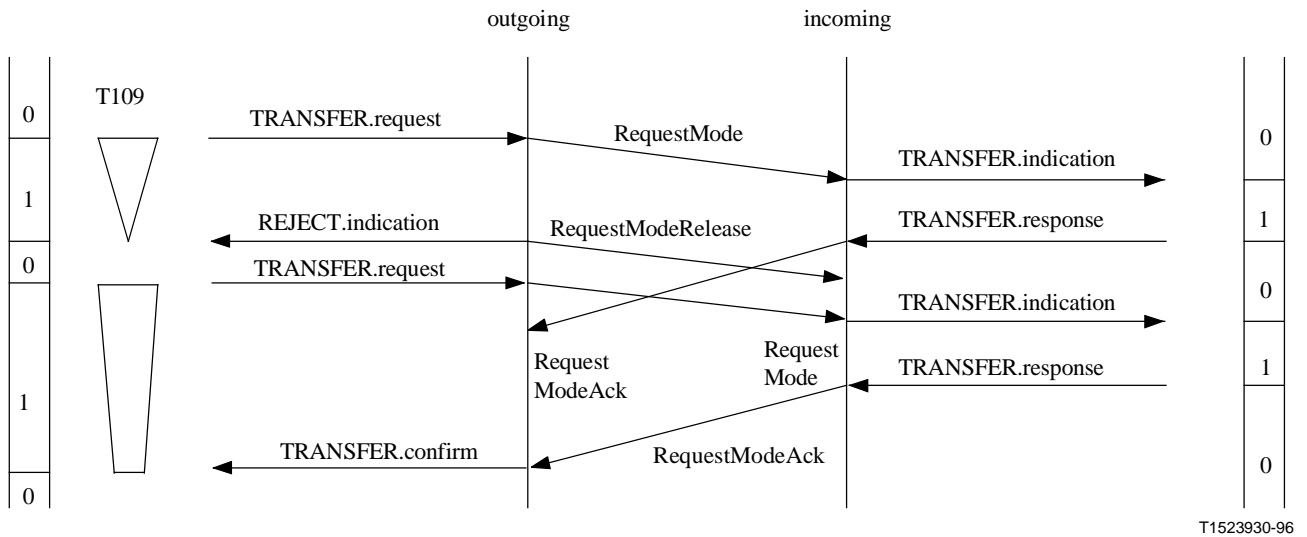
NOTE – The first request was unsuccessful.

Figure II.7-3/H.245 – Mode request with a second mode request before acknowledgement of the first request



NOTE – The mode request was unsuccessful.

Figure II.7-4/H.245 – Mode request with timer T109 expiry



NOTE – The first RequestModeAck message is ignored at the outgoing MRSE. Only the second request was successful.

Figure II.7-5/H.245 – Mode request with timer T109 expiry followed by a second mode request

II.8 Round Trip Delay Signalling Entity

The following figures illustrate RTDSE procedures. The RTDSE states of IDLE and AWAITING RESPONSE are labelled as "0" and "1" respectively.

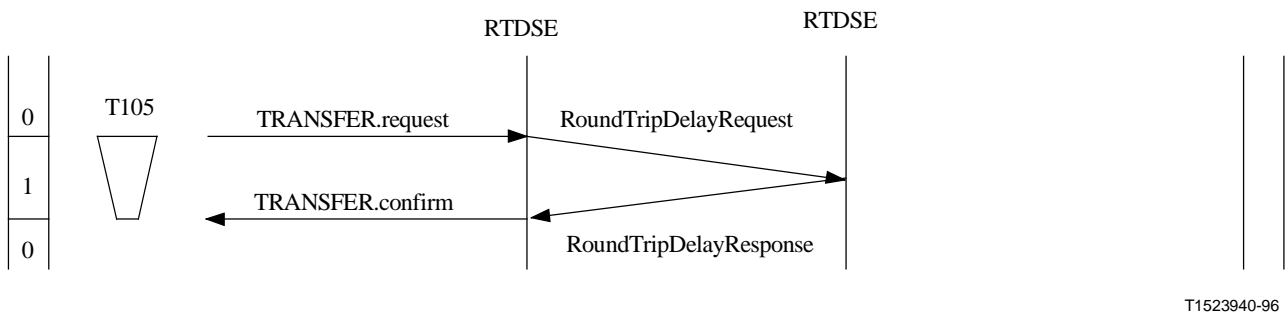


Figure II.8-1/H.245 – Round trip delay determination procedure

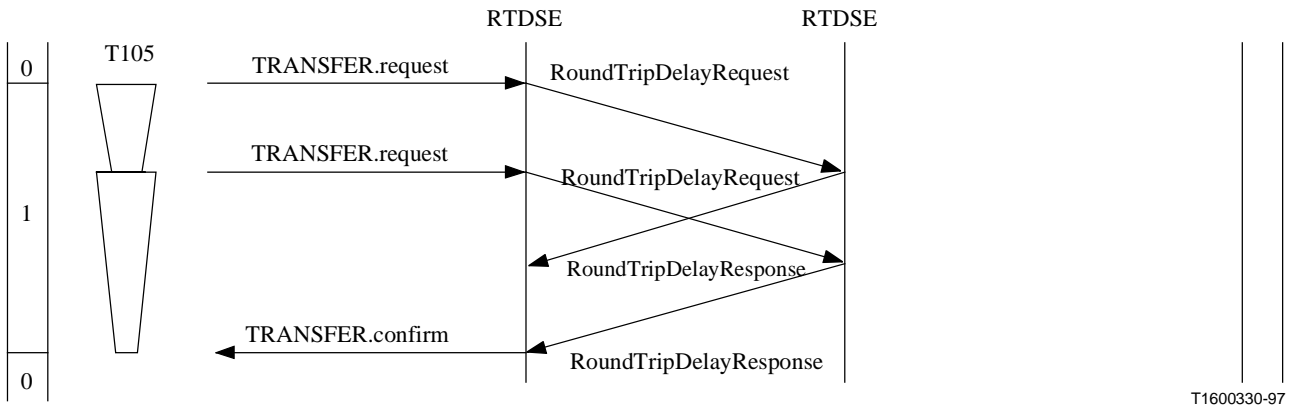


Figure II.8-2/H.245 – Round trip delay determination procedure with an earlier unacknowledged round trip delay procedure outstanding

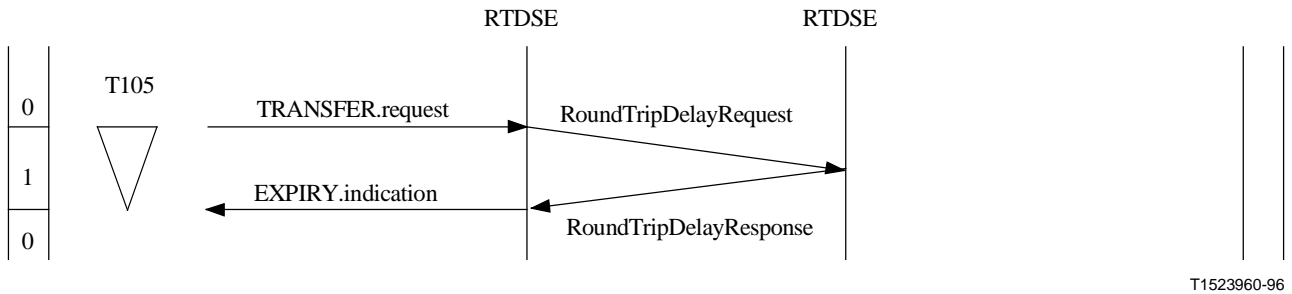
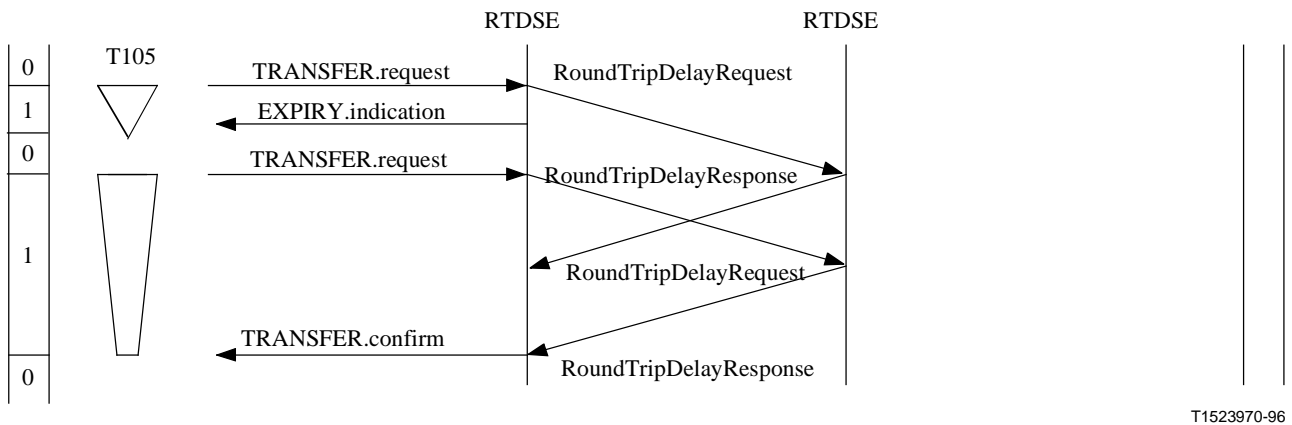


Figure II.8-3/H.245 – Round trip delay determination procedure with timer T105 expiry



NOTE – The RoundTripDelayResponse message from the first procedure arrives during the second procedure and is ignored.

Figure II.8-4/H.245 – Round trip delay determination procedure with timer T105 expiry, followed by a second round trip delay determination procedure

II.9 Bidirectional Logical Channel Signalling Entity

The following figures illustrate B-LCSE procedures. The outgoing B-LCSE states of RELEASED, AWAITING ESTABLISHMENT, ESTABLISHED, and AWAITING RELEASE are labelled as "0", "1", "2", and "3" respectively. The incoming B-LCSE states of RELEASED, AWAITING ESTABLISHMENT, AWAITING CONFIRMATION, and ESTABLISHED, are labelled as "0", "1", "2", and "3" respectively.

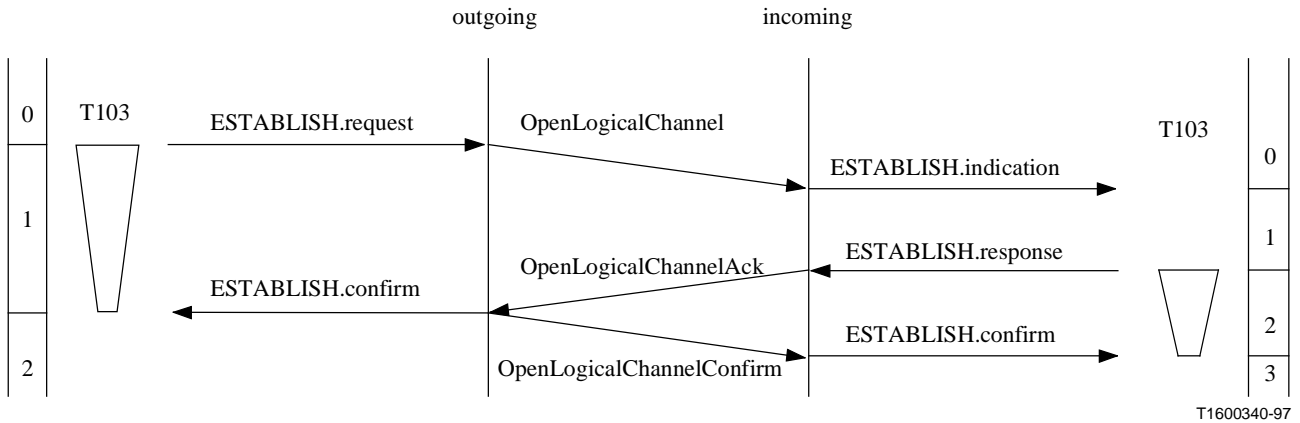


Figure II.9-1/H.245 – Bidirectional logical channel establishment

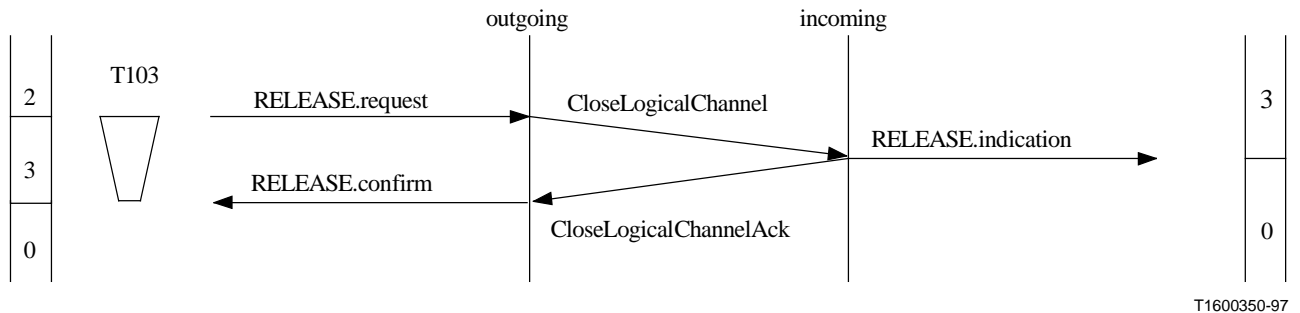


Figure II.9-2/H.245 – Bidirectional logical channel release

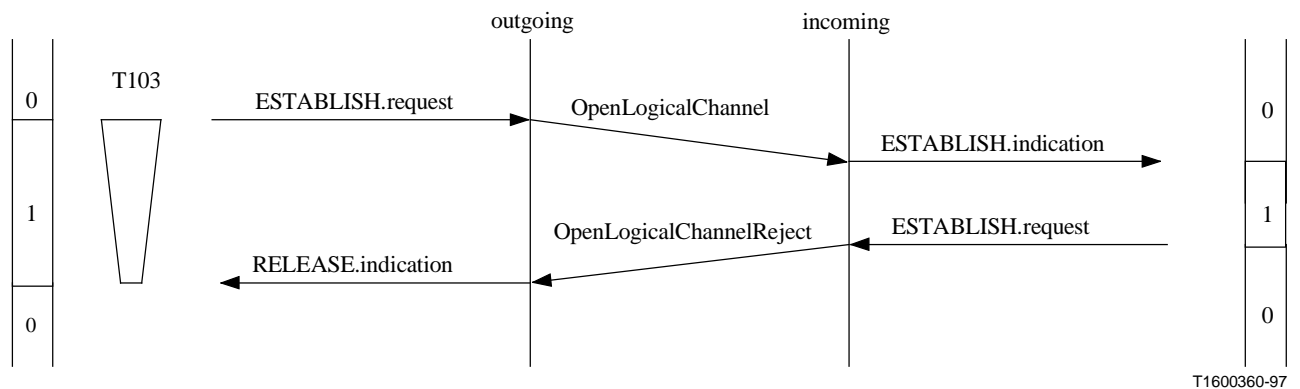


Figure II.9-3/H.245 – Bidirectional logical channel establishment rejection by peer B-LCSE user

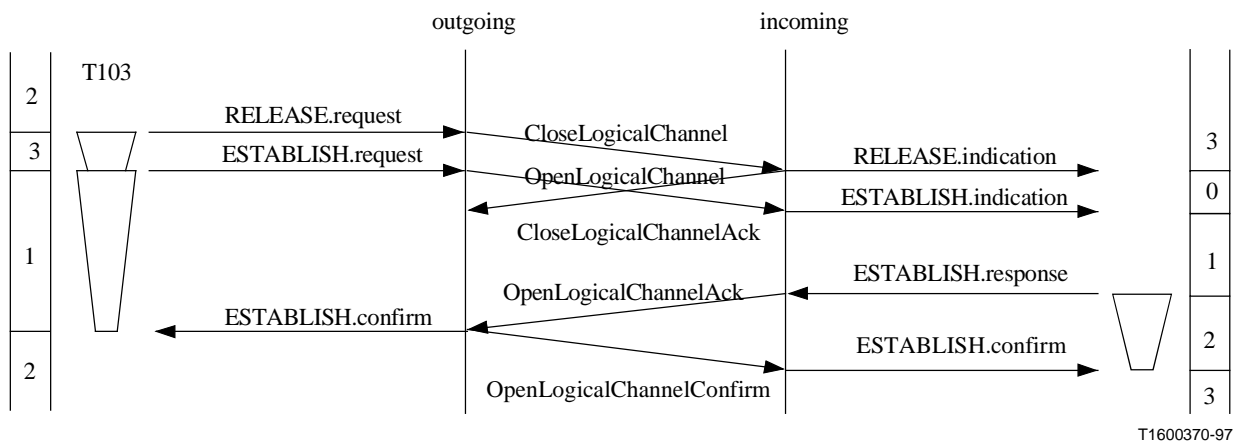


Figure II.9-4/H.245 – Bidirectional logical channel release followed by immediate re-establishment

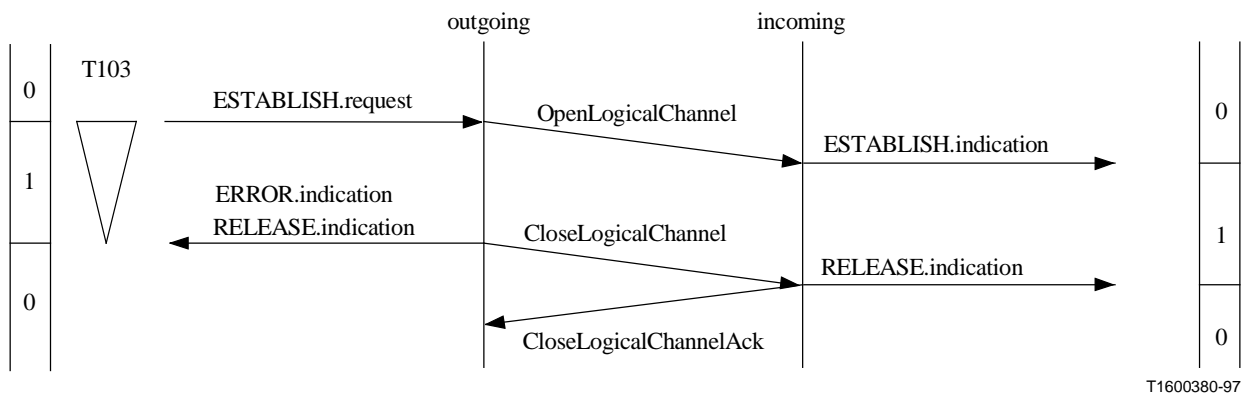
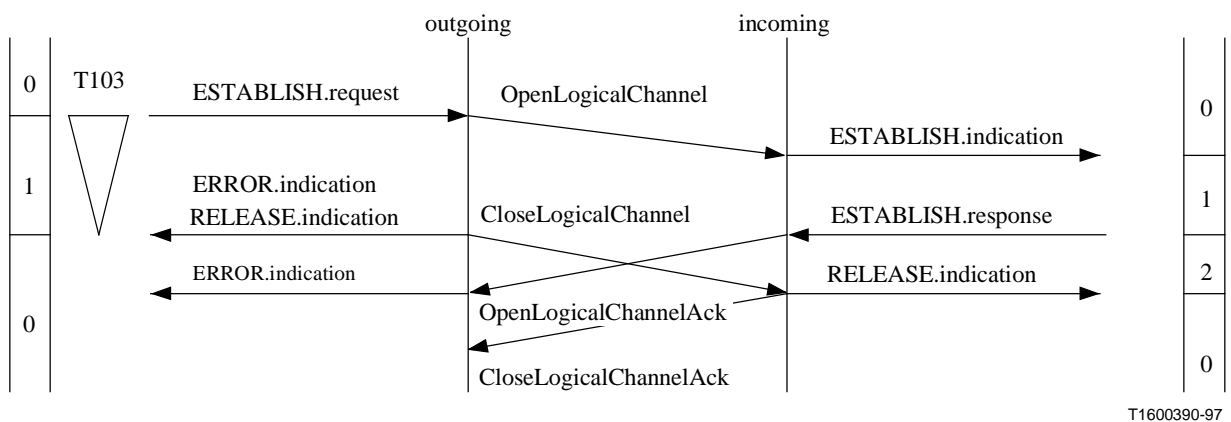


Figure II.9-5/H.245 – Bidirectional logical channel establishment request with expiry of timer T103 at the outgoing side due to slow response from peer incoming B-LCSE user



NOTE – Timer T103 at the outgoing side has expired after transmission of the OpenLogicalChannelAck message at the incoming B-LCSE, but before reception of the OpenLogicalChannelAck message at the outgoing B-LCSE.

Figure II.9-6/H.245 – Bidirectional logical channel establishment request with expiry of timer T103 at the outgoing side

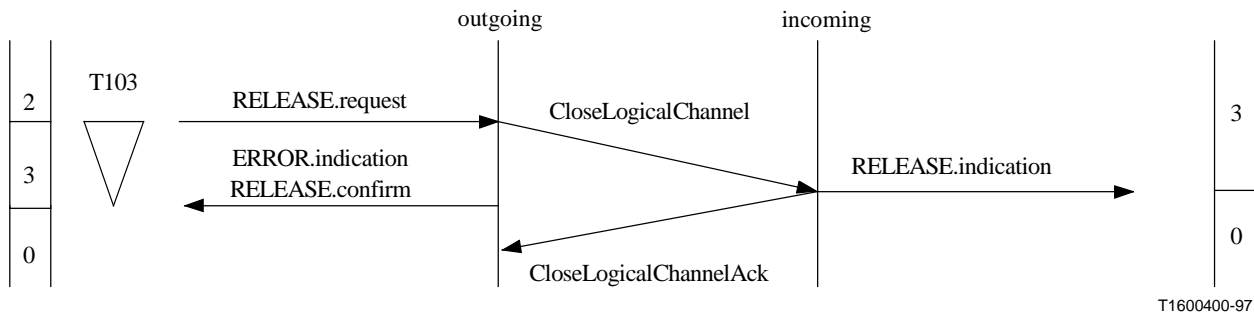


Figure II.9-7/H.245 – Bidirectional logical channel release request with expiry of timer T103 at the outgoing side

APPENDIX III

Summary of procedure timers and counters

This appendix provides a list of the timers and counters specified in clause 8.

This Recommendation does not define the values loaded into these timers. The values may be defined in other Recommendations such as H.310, H.323 and H.324.

III.1 Timers

Table III.1 shows the timers specified in this Recommendation.

Table III.1/H.245 – Procedure Timers

Timer	Procedure	Definition
T106	Master Slave Determination	This timer is used in the OUTGOING AWAITING RESPONSE state and during the INCOMING AWAITING RESPONSE state. It specifies the maximum time during which no acknowledgement message may be received.
T101	Capability Exchange	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no TerminalCapabilitySetAck or TerminalCapabilitySetReject message may be received.
T103	Unidirectional and Bidirectional Logical Channel Signalling	This timer is used in the AWAITING ESTABLISHMENT and AWAITING RELEASE states. It specifies the maximum time during which no OpenLogicalChannelAck or OpenLogicalChannelReject or CloseLogicalChannelAck message may be received.
T108	Close Logical Channel	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no RequestChannelCloseAck or RequestChannelCloseReject message may be received.
T104	H.223 Multiplex Table	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no MultiplexEntrySendAck or MultiplexEntrySendReject message may be received.
T109	Mode Request	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no RequestModeAck or RequestModeReject message may be received.

Table III.1/H.245 – Procedure Timers (concluded)

Timer	Procedure	Definition
T105	Round Trip Delay	This timer is used in the AWAITING RESPONSE state. It specifies the maximum time during which no RoundTripDelayResponse message may be received.
T107	Request Multiplex Entry	This timer is used during the AWAITING RESPONSE state. It specifies the maximum time during which no RequestMultiplexEntryAck or RequestMultiplexEntryReject message may be received.
T102	Maintenance Loop	This timer is used during the AWAITING RESPONSE state. It specifies the maximum allowed time during which no MaintenanceLoopAck or MaintenanceLoopReject message may be received.

III.2 Counters

Table III.2 shows the counters specified in this Recommendation.

Table III.2/H.245 – Procedure Counters

Timer	Procedure	Definition
N100	Master slave determination	This counter specifies the maximum number of times that MasterSlaveDetermination messages will be sent during the OUTGOING AWAITING RESPONSE state.

APPENDIX IV

H.245 Extension Procedure

This Recommendation is a "living document" used by a number of systems Recommendations including H.310, H.323, H.324, and V.70, which is expected to be extended, in a backward-compatible way, likely at each meeting of ITU-T Study Group 16. This appendix explains the procedure that should be used to add extensions to this Recommendation.

At a given point in time there is only one H.245 syntax in force. No other ITU-T Recommendation should include other variants of H.245 syntax in their Recommendations in a normative manner.

Requests for extensions to this Recommendation should be submitted as a White Contribution or formal liaison to Study Group 16, with a copy sent as early as possible to the ITU-T Rec. H.245 Rapporteur and editor. Such requests should include:

- 1) Functional requirements for syntax to be drafted by the ITU-T Rec. H.245 editor or proposed syntax based on the current approved version of ITU-T Rec. H.245.
- 2) Proposed semantics for clause 7.
- 3) Proposed procedures for clause 8 if new procedures are requested.

All extensions to ITU-T Rec. H.245 must be backwards compatible with all previous versions of Recommendation H.245. Pre-existing syntax, semantics, and procedures cannot be changed. The meaning of pre-existing syntax cannot be changed. Specifically, when an H.245 Capability is

extended, the extension shall not change the meaning of the original capability in such a way that a terminal which does not understand the extension would need to modify its operation to use the capability without the extension. All ASN.1 extension components should be constrained.

Requests should be submitted as early as possible to allow time for review of extensions by H.245 experts in Study Group 16. It must be understood that the exact requested syntax may be modified because of:

- 1) Verification of correct ASN.1 syntax
- 2) Harmonization with other, conflicting, requests for H.245 extensions
- 3) Backward compatibility with pre-existing versions of H.245
- 4) Expert review of placement of new functions relative to the existing H.245 structure
- 5) Naming that is inconsistent with pre-existing syntax
- 6) Unconstrained or ambiguous ASN.1 components

Abbreviations and acronyms should be avoided, especially if a word or phrase is not abbreviated or expressed as an acronym in pre-existing syntax. For example, the word, Parameters, should not be abbreviated as Params. If a word has been used in pre-existing syntax, do not use another word with the same meaning. For example, call components of an aggregate type, Entry, instead of Item, because Entry has consistently been used to describe this. Be consistent.

Although all ASN.1 components should be constrained, how to constrain the most common types is described below.

Constrain SET OF and SEQUENCE OF ASN.1 components by providing either a minimum and maximum or a fixed size. If there is no inherent maximum based upon a component's semantics, choose a reasonable, although arbitrary, maximum such as 256. If a SET OF or SEQUENCE OF component is OPTIONAL, specify a non-zero minimum unless there is a semantic difference between the cases, present-but-empty and not present, in which case the semantic difference should be described. If a request for extensions contains SET OF or SEQUENCE OF components that are not constrained, the editor may use SIZE (1..256) as a default constraint.

Constrain ASN.1 character string components by providing a size, either a minimum and maximum or a fixed size. If a request for extensions contains character string components that are not constrained, the editor may use SIZE (0..255) as a default constraint.

Constrain INTEGER components by providing a range of values. If there is no inherent range based upon a component's semantics, choose a reasonable, although arbitrary, range whose maximum value is chosen from the following:

255	$(2^8 - 1)$
65535	$(2^{16} - 1)$
16777215	$(2^{24} - 1)$
4294967295	$(2^{32} - 1)$

If a request for extensions contains INTEGER components that are not constrained, the editor may use INTEGER (0..4294967295) as a default.

The H.245 editor will review all extension requests and propose final text for extended versions of ITU-T Rec. H.245 for Study Group 16 approval by the Resolution 1 process. Upon Study Group approval of each new version of ITU-T Rec. H.245, the H.245 version number in **protocolIdentifier** will be incremented to identify the new version.

Please note that it is the intention of Study Group 16 to accept only harmonized H.245 extensions originating from the H.245 editor.

APPENDIX V

The replacementFor Procedure

The H.245 **replacementFor** procedure allows seamless changing of modes from one codec to another without the need for two media decoders. This procedure may be used only if the receiving terminal has indicated the **maxPendingReplacementFor** capability.

Since opening and closing of H.245 logical channels is not synchronized with media content, media dropout can occur between the time of closing a logical channel and the opening of its replacement. The **replacementFor** parameter allows the avoidance of such media dropout.

Example

Suppose logical channel 723 is open, carrying G.723.1 audio, and it is desired to switch to G.711 (on logical channel 711), but the receiver has a capability for only one audio channel. The **replacementFor** procedure may be used by the transmitter to effect a seamless mode change as follows:

- 1) *Only for the case of H.323 using RSVP*, since the new channel will require more bandwidth (64 kbit/s) than the existing channel (6.4 kbit/s), the transmitter and receiver establish a larger RSVP bandwidth reservation.
- 2) The transmitter sends **OpenLogicalChannel** for the new logical channel 711, including the **replacementFor** parameter, referring to the existing logical channel 723.
This tells the receiver that logical channel 711 is a *replacement for* logical channel 723, and that logical channel 711 will never carry traffic simultaneously with logical channel 723.
- 3) While continuing to decode G.723.1 from logical channel 723, the receiver prepares for a seamless switch to decoding G.711.

Such preparation might include loading appropriate decoder software.

When the receiver has completed preparations to accept the G.711 audio stream, it responds with **OpenLogicalChannelAck** for logical channel 711. For H.323, the media and media control transport addresses returned are the same as those already used for logical channel 723.

- 4) The transmitter stops sending G.723 audio on logical channel 723 and seamlessly begins sending G.711 audio on logical channel 711.
- 5) The transmitter immediately sends **CloseLogicalChannel** for logical channel 723, as this logical channel is no longer carrying any traffic, and is no longer needed.
- 6) *Only for the case of H.323 using RSVP*, if the new channel requires less bandwidth than the original channel, the transmitter and receiver establish a smaller RSVP bandwidth reservation (does not apply in this example).

In all cases, LCSE and B-LCSE operations conform to normal procedures. The **replacementFor** parameter merely informs the receiver of the pending mode change and that the two logical channels will not be used simultaneously, and therefore that the second logical channel can (in some implementations) be accepted in cases where it would otherwise be rejected (for lack of the capability to receive another independent logical channel).

Note that in some cases the receiver may reject the attempt to open the logical channel using the **replacementFor** mechanism (for example, if a receiver can accept the **replacementFor** mechanism for audio channels, but not for video channels). In that case transmitters should re-try the mode change without **replacementFor**, for example by closing the channel, then opening a new one, accepting any temporary media dropout.

Note also that in H.323 systems, receivers are required to re-use the existing media and media control transport addresses. The switch over point to the new logical channel is marked by the RTP header.

APPENDIX VI

Examples of H.263 Capability Structure Settings

To clarify the usage of the H.263 Capability Structure, a number of examples are given in this appendix.

VI.1 Examples of Enhancement Layer H.245 parameter setting

Table VI.1 shows the following example settings of parameters of enhancement layer parameters.

Example # 1: this signals a simple H.263 base video capability at 10 frames/s, maximum bit rate of 20 kbit/s with no options.

Example # 2: these parameters settings signal the capability of a logical channel stream with a spatial enhancement layer at QCIF resolution, 10 frames/s at a maximum bit rate of 5 kbit/s and no other options set.

Example # 3: these parameters settings signal the capability of a logical channel stream with a SNR enhancement layer at SQCIF resolution, 10 frames/s at a maximum bit rate of 5 kbit/s and no other options set.

Example # 4: these parameters settings signal the capability of a logical channel stream with a three enhancement layers. Two SNR enhancement layers, one at SQCIF the other a QCIF, at 10 frames/s and no other options set and the other a spatial enhancement layer at CIF resolution, 10 frames/s and no other options set; all three combined at a maximum bit rate of 15 kbit/s.

Example # 5: these parameters settings signal the capability of a logical channel stream with a three enhancement layers and a base layer at a maximum bit rate of 25 kbit/s. The base layer in QCIF with no options. In addition the terminal is capable of one SNR enhancement layer at QCIF, 10 frames/s and no other options set, one SNR enhancement layer at CIF resolution, 10 frames/s with no other options set and a spatial enhancement layer at CIF resolution, 10 frames/s and no other options set.

Table VI.1/H.245 – Enhancement Layer H.245 parameter setting examples

H263Capability parameter		Examples								
		1	2	3	4			5		
sqcifMPI		3	NP	NP	NP			NP		
qcifMPI		NP	NP	NP	NP			3		
cifMPI		NP	NP	NP	NP			NP		
cif4MPI		NP	NP	NP	NP			NP		
cif16MPI		NP	NP	NP	NP			NP		
maxBitRate		200	50	50	150			250		
unrestrictedVector		F	F	F	F			F		
arithmeticCoding		F	F	F	F			F		
advancedPrediction		F	F	F	F			F		
pbFrames		F	F	F	F			F		
temporalSpatialTradeOffCap		F	F	F	F			F		
hrd-B		NP	NP	NP	NP			NP		
bppMaxKb		NP	NP	NP	NP			NP		
slowSqcifMPI		NP	NP	NP	NP			NP		
slowQcifMPI		NP	NP	NP	NP			NP		
slowCifMPI		NP	NP	NP	NP			NP		
slowCif4MPI		NP	NP	NP	NP			NP		
slowCif16MPI		NP	NP	NP	NP			NP		
errorCompensation		NP	NP	NP	NP			NP		
SET OF (EnhancementOptions*) =		NP	NP	1	1	2		1	2	
snrEnhancement	sqcifMPI			3	3	NP		NP	NP	
	qcifMPI			NP	NP	3		3	NP	
	cifMPI			NP	NP	NP		NP	3	
	cif4MPI			NP	NP	NP		NP	NP	
	cif16MPI			NP	NP	NP		NP	NP	
	maxbitrate			50	50	50		50	50	
	SET OF (EnhancementOptions*) =	NP	1	NP	NP	NP	1	NP	NP	1
spatialEnhancement	sqcifMPI		NP				NP			NP
	qcifMPI		3				NP			NP
	cifMPI		NP				3			3
	cif4MPI		NP				NP			NP
	cif16MPI		NP				NP			NP
	maxbitrate		50				50			50

Table VI.1/H.245 – Enhancement Layer H.245 parameter setting examples (concluded)

H263Capability parameter		Examples								
		1	2	3	4			5		
SET OF (EnhancementOptions*) =		NP	NP	NP	NP	NP	NP			
bframeEnhancement	sqcifMPI									
	qcifMPI									
	cifMPI									
	cif4MPI									
	cif16MPI									
	maxbitrate									
NP = Not Present, T = TRUE, F = FALSE										
*other options below "maxbitrate" in the EnhancementOptions structure not shown.										

VI.2 Examples of Video Back Channel H.245 parameter setting

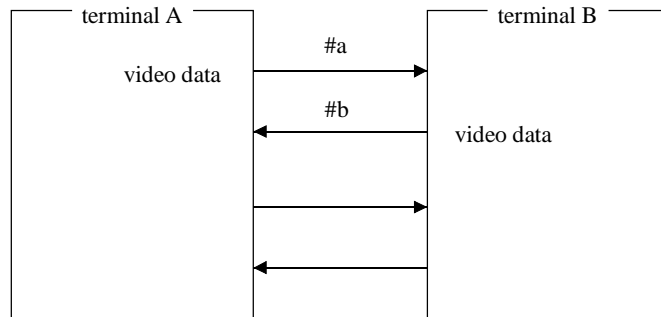
This subclause provides examples of H263Capability and H263Options settings for video back-channel operation.

Example 1: Separate Logical Channel mode

In this mode, an extra bidirectional logical channel is opened for video back-channel messages. The dependency between a forward video channel and the video back-channel is described by **forwardLogicalChannelDependency** and **reverseLogicalChannelDependency** in the OpenLogicalChannel message.

The logical channel for video back-channel messages shall only be established after the forward video channel is established. If an OpenLogicalChannel message is received with a dependency reference to a non-existing channel, the terminal shall respond with an OpenLogicalChannelReject with the reason code invalidDependentChannel. An example follows.

- 1) A bidirectional logical channel for video data is opened between terminal A and terminal B as shown in Figure VI.1. The OpenLogicalChannel message for the bidirectional logical channel includes RefPictureSelectionCapability in H263VideoCapability.



T1604350-97

Figure VI.1/H.245 – Logical Channels for Video Data

- 2) Next, a bidirectional logical channel for video back-channel messages is opened, as shown in Figure VI.2. In this example, we assume that terminal A requests to open the bidirectional logical channel. (If the terminal B requests to open the channel, forwardLogicalChannelDependency is replaced by reverseLogicalChannelDependency and vice versa.) The OpenLogicalChannel message of this logical channel includes forwardLogicalChannelDependency in forwardLogicalChannelParameters indicating LCN of LC #a in Figure VI.2 and reverseLogicalChannelDependency in reverseLogicalChannelParameters indicating LCN of LC #b, as well as separateVideoBackChannel.

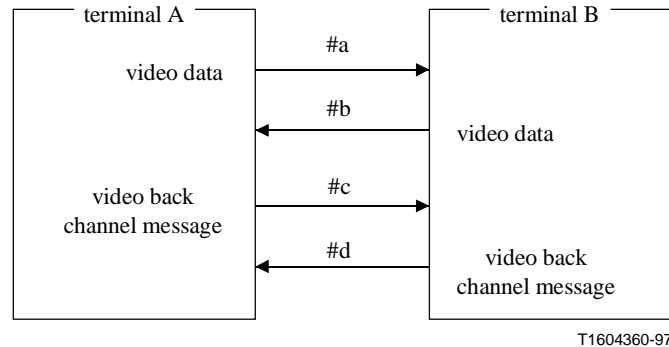


Figure VI.2/H.245 – Logical Channels for Separate Logical Channel Mode

- 3) After the logical channel for video back-channel messages is established, terminal A sends the video data to LC #a and receives from LC #d, the video back-channel messages that correspond to the video data sent to LC #a. In the same manner, terminal A receives the video data from LC #b and sends to LC #c, the video back-channel messages that correspond to the video data from terminal B.

An example of setting H263Capability parameters in each OpenLogicalChannel messages is summarized in Table VI.2. Only a part of capabilities of H263Capability is shown for simplicity.

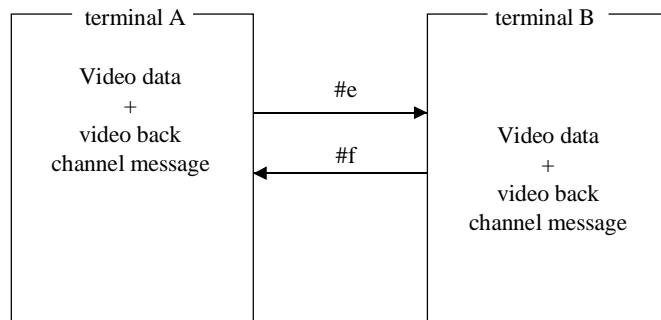
**Table VI.2/H.245 – H263Capability setting examples
for OpenLogicalChannel messages**

H263Capability parameter	H263Capability in OpenLogicalChannel message		
	#a, #b	#c, #d	#e, #f
sqcifMPI	NP	NP	NP
qcifMPI	3	NP	3
cifMPI	NP	NP	NP
cif4MPI	NP	NP	NP
cif16MPI	NP	NP	NP
maxBitRate	240	10	240
refPictureSelection		NP	
additionalPictureMemory	unspecified	–	unspecified
videoMuxCapability	False	–	(shall be) True
videoBackChannelSendCapability	ackAndNackMessage	–	ackAndNackMessage
separateVideoBackChannel	False	True	False
NP: Not Present			

Example 2: VideoMux mode

When a terminal indicates the videoMuxCapability in RefPictureSelectionCapability during capability exchange, another terminal may use this mode to send video back-channel messages. Because video back-channel messages are multiplexed into the coded video bitstream, the terminals do not need to establish an extra logical channel for video back-channel messages. An example follows.

- 1) A bidirectional logical channel for video is opened by the OpenLogicalChannel message including refPictureSelectionCapability with the true values VideoMux mode in their H263VideoCapability. (See Figure VI.3.)
- 2) After the logical channel for video is established, terminal A sends the video data to LC #e and receives from LC #f, the video back-channel messages that correspond to the video data sent to LC #e are multiplexed into the video data from terminal B.



T1604370-97

**Figure VI.3/H.245 – Logical Channels
for VideoMux Mode**

Example 3: Separate Logical Channel mode in unidirectional video communication

This example shows the case when only terminal A sends video data and terminal B sends only video back-channel messages. (Figure VI.4.) The parameter settings of logical channel #g and #h are shown in Table VI.3.

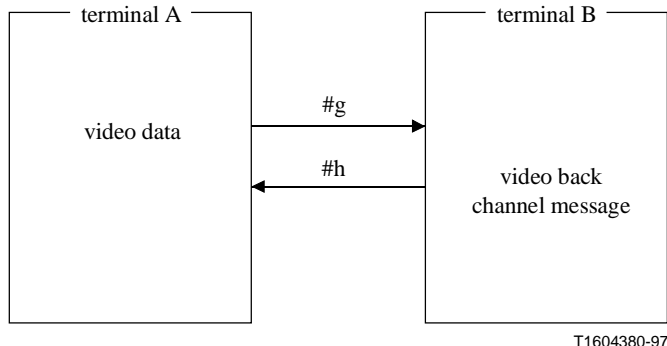


Figure VI.4/H.245 – Separate Logical Channel mode in unidirectional video communication

Example 4: Coexistence of Separate Logical Channel mode with VideoMux mode

The example illustrated in Figure VI.5 shows the case when only terminal A uses the Separate Logical Channel mode to receive video back-channel messages from terminal B via LC #l and terminal B uses the VideoMux mode to receive video back-channel message via LC #i. This example may not be realistic but is a possible configuration. The parameter settings of each logical channels are shown in Table VI.3.

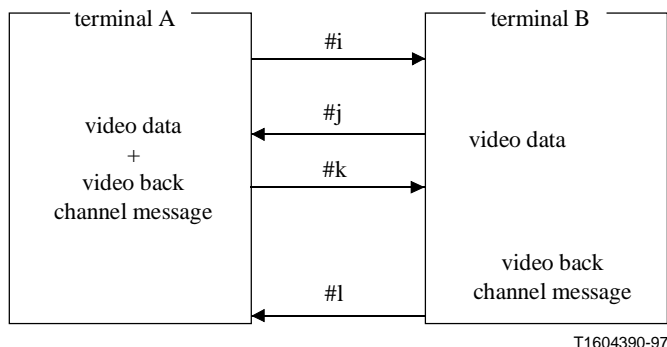


Figure VI.5/H.245 – Coexistence of Separate Logical Channel mode with VideoMux mode

Table VI.3/H.245 – H263Capability setting examples for OpenLogicalChannel messages

H263Capability parameter	H263Capability in OpenLogicalChannel message					
	#g	#h	#i	#j	#k	#l
sqcifMPI	NP	NP	NP	NP	NP	NP
qcifMPI	3	NP	3	3	NP	NP
cifMPI	NP	NP	NP	NP	NP	NP
cif4MPI	NP	NP	NP	NP	NP	NP
cif16MPI	NP	NP	NP	NP	NP	NP
maxBitRate	240	10	240	240	10	10
refPictureSelection		NP			NP	NP
additionalPictureMemory	unspecified	–	unspecified	unspecified	–	–
videoMuxCapability	F	–	F	(shall be) T	–	–
videoBackChannelSend Capability	ackAndNack Message	–	ackAndNack Message	ackAndNack Message	–	–
separateVideoBackChannel	F	T	F	F	F	T
NP = Not Present, T = True, F = False						

ITU-T RECOMMENDATIONS SERIES

- Series A Organization of the work of the ITU-T
- Series B Means of expression: definitions, symbols, classification
- Series C General telecommunication statistics
- Series D General tariff principles
- Series E Overall network operation, telephone service, service operation and human factors
- Series F Non-telephone telecommunication services
- Series G Transmission systems and media, digital systems and networks
- Series H Audiovisual and multimedia systems**
- Series I Integrated services digital network
- Series J Transmission of television, sound programme and other multimedia signals
- Series K Protection against interference
- Series L Construction, installation and protection of cables and other elements of outside plant
- Series M TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
- Series N Maintenance: international sound programme and television transmission circuits
- Series O Specifications of measuring equipment
- Series P Telephone transmission quality, telephone installations, local line networks
- Series Q Switching and signalling
- Series R Telegraph transmission
- Series S Telegraph services terminal equipment
- Series T Terminals for telematic services
- Series U Telegraph switching
- Series V Data communication over the telephone network
- Series X Data networks and open system communications
- Series Y Global information infrastructure
- Series Z Languages and general software aspects for telecommunication systems